Stereo-Camera–LiDAR Calibration for Autonomous Driving

Eugeniu Vezeteu

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 05.07.2021

Thesis supervisor:

Prof. Simo Särkkä

Thesis advisors:

M.Sc. Petri Manninen

M.Sc. Heikki Hyyti

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Eugeniu Vezeteu

Title: Stereo-Camera–LiDAR Calibration for Autonomous Driving

Date: 05.07.2021 Language: English Number of pages: 7+62

International double degree programme ICT Innovation, Autonomous Systems

Supervisor: Prof. Simo Särkkä

Advisors: M.Sc. Petri Manninen, M.Sc. Heikki Hyyti

Perception is one of the key factors to successful self-driving. According to recent studies in developing perception 3D range scanners combined with stereo camera vision are the most utilized sensors in autonomous vehicle perception systems. To enable accurate perception, the sensors must be calibrated before the sensor data can be fused. Calibration minimizes measurement errors caused by the nonidealities of individual sensors and errors caused by the transformation between different sensor frames.

This thesis presents camera-LiDAR calibration, synchronisation, and data fusion techniques. It can be argued that the quality of data is more important to the calibration than the actual optimization algorithms, therefore, one challenge addressed in this thesis is accurate data collection with different calibration targets and result validation with different optimization algorithms.

We estimated the vehicle windshield effect on camera calibration and show that the error caused by the windshield can be decreased by using more complex distortion models than the standard model. Synchronisation is required to ensure that sensors provide measurements at the same time. The sensor data used in this thesis was synchronized by using an external trigger signal from a GNSS receiver. The camera-LiDAR extrinsic calibration was performed using synchronised 3D-2D (LiDAR points and camera pixels) and 3D-3D (LiDAR points and stereo camera) point correspondences. This comparison demonstrates that the best method to estimate camera-LiDAR extrinsic parameters is to use 3D-2D point correspondences. Moreover, a comparison between camera-based and LiDAR 3D reconstruction is presented. Due to different sensors viewpoint, some data points are occluded, therefore, we propose a camera-LiDAR occlusion handling algorithm to remove occluded points. The quality of the calibration is demonstrated visually, by fusing and aligning the LiDAR point cloud and the image.

Keywords: Camera, LiDAR, calibration, synchronisation, stereo vision, 3D reconstruction, windshield, point cloud.

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Eugeniu Vezeteu

Työn nimi: Camera-LiDAR kalibrointi autonomiselle ajamiselle

Päivämäärä: 05.07.2021 Kieli: Englanti Sivumäärä: 7+62

Kansainvälinen kaksoistutkinto-ohjelma ICT-innovaatio, Autonomiset Järjestelmät

Työn valvoja: Prof. Simo Särkkä

Työn ohjaajat: M.Sc. Petri Manninen, M.Sc. Heikki Hyyti

Havainnointi on yksi onnistuneen itseajon avaintekijöistä ja viimeaikaisen tutkimuksen mukaan 3D-etäisyysmittaus yhdistettynä stereokameranäköön on autonomisten autojen havainnointijärjestelmissä yleisimmin käytetty sensorikokoonpano. Mahdollisimman tarkan havainnoinnin mahdollistamiseksi sensorit pitää kuitenkin kalibroida ennen sensoridatan yhdistämistä. Kalibroinnilla minimoidaan yksittäisten sensoreiden epäideaalisuudesta johtuvaa mittausvirhettä ja sensoreiden välisestä siirtymästä ja kierrosta johtuvaa virhettä

Työssä esitetään kameran ja 3D-laserkeilaimen kalibrointi, datan synkronointi sekä datan yhdistämistekniikka. Voidaan väittää, että kalibrointiin käytettävän datan tarkkuus on lopputuloksen kannalta olennaisempaa kuin itse kalibrointiin käytettävä optimointialgoritmi. Yksi tämän työn haasteista onkin ollut kerätä mahdollisimman tarkka data-aineisto eri kalibrointikohteilla ja varmistaa kalibroinnin lopputulos eri optimointialgoritmeilla.

Työssä on arvioitu tuulilasin vaikutusta auton sisään asennettujen kameroiden kalibrointiin ja esitetty miten tuulilasin aiheuttamaa virhettä voidaan pienentää huomattavasti käyttämällä vakiomallia monimutkaisempia kalibrointimalleja. Sensoreiden synkronointi pitää varmistaa, jotta mittausten ajanhetki tunnetaan ja eri sensoreiden data on yhdistettävissä. Työssä käytetty data on synkronoitu laukaisemalla sensorit ulkoisesti satelliittipaikannusjärjestelmän vastaanottimen avulla. Kameran ja laserkeilaimen välinen siirtymä ja kierto on määritetty käyttämällä sekä 3D-2D (laserkeilatut pisteet ja kuvapikselit) että 3D-3D (laserkeilatut pisteet ja stereokamerapisteet) pistevastaavuuksia. Vertailu osoittaa, että 3D-2D pistevastaavuuksien käyttö on parempi kameran ja laserkeilaimen välisen siirtymän ja kierron arviointiin. Lisäksi esitetään sekä stereokameraan että 3D-laserkeilaimeen perustuvan 3D rekonstruktion vertailu. Sensoreiden eri katselukulmasta johtuen jotkin pisteet saattavat todellisuudessa jäädä katvesseen, eikä niitä voi yhdistää toisen sensorin dataan. Työssä on kehitetty katvepisteiden havaitsemiseen ja poistamiseen tarkoitettu algoritmi. Kalibroinnin laatua on havainnollistettu visuaalisesti yhdistämällä laserkeilaimen ja kameran tuottama data.

Avainsanat: Kamera, LiDAR, kalibrointi, synkronointi, stereonäkymä, 3D-rekonstruktio, tuulilasi, pistepilvi.

Preface

I have always been interested in Computer Vision and Robotics. Finnish Geospatial Research Institute (FGI) allowed me to apply my knowledge in this area, especially in the field of self-driving cars. I think that driverless revolution will change the world and I am proud to contribute to it. This research was realized at the FGI in the department of Remote Sensing and Photogrammetry, as part of the autonomous driving car research project.

I want to express my deepest respect to my advisers Petri Manninen and Heikki Hyyti, for their advice and feedback that has always pushed me towards the right solution. I appreciate the time and patience with which Petri Manninen instructed me. Sincere thanks to Heikki Hyyti, who listened to my interests and proposed this topic to me. I am also very grateful to my supervisor Professor Särkkä Simo, for constructive feedback and guidance on this thesis. I also want to thank Professor Juha Hyyppä for the work he did to make this project possible and for hiring me. I would like to express my sincere thanks to the autonomous driving research team: Paula Litkey, Jyri Maanpää, Carlos Tiana Gómez, Josef Taher. I am deeply grateful for the pleasant and encouraging work atmosphere. Last but not least, I would like to thank my family for their constant support.

Otaniemi, 05.07.2021

Eugeniu Vezeteu

Contents

Symbols and abbreviations

Symbols

$\mathbf{E}$	Essential matrix
$\mathbf{F}$	Fundamental matrix
f	Focal length in pixels
f_x	Focal length in pixels in x direction
f_y	Focal length in pixels in y direction
$\mathbf{H}$	Homography transformation matrix
$\mathbf{K}$	Internal camera calibration matrix
$\mathbf{P}$	Projection matrix
p_x	Optical center in pixels, x coordinate
p_y	Optical center in pixels, y coordinate
$\mathbf{R}$	Rotation matrix
s_k	skewness parameter
$\mathbf{t}$	Translation vector
$\mathbf{X}$	3D point
$\mathbf{x}$	2D point

Abbreviations

ADAS	Advanced Driver Assistance Systems
ARVO	Autonomous Research Vehicle Observatory
CMP	Complete distortion model
DOF	Degree of Freedom
FGI	Finnish Geospatial Research Institute
FOV	Field of View
GNSS	Global Navigation Satellite System
ICP	Iterative closest point
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
MLS	Mobile Laser Scanner
PnP	Perspective-n-Point
PPS	Pulse Per Second
RANSAC	RANdom SAmple Consensus
RAT	Rational distortion model
ROS	Robot Operating System
RMS	Root Means Squared error
SLAM	Simultaneous Localization and Mapping
TIL	Tilted distortion model
THP	Thin Prism distortion model

1 Introduction

The rapid development of the autonomous vehicle sector has resulted in increased demand for positioning and tracking systems. Paden et al. (2016) argues that the self-driving feature is mainly composed of 2 parts: the perception and the decision-making systems. To improve autonomous perception and navigation, self-driving cars are embedded with a varied set of sensors. Inertial measurement unit (IMU), light detection and ranging (LiDAR), cameras, and global navigation satellite system (GNSS) receivers are among the most commonly used. These types of sensors differ not only by price but also by the type of information that they provide to the system. For example, IMU provides information about gravity, acceleration, and magnetic field, LiDAR creates point cloud of the environment, and GNSS gives positioning data. As Luettel et al. (2012) stated, an estimate based on data obtained from a single sensor is not sufficient to perform robust and accurate estimation. Although these sensors work independently of each other, none of them offer long-term solutions on their own. IMU may drift on long trajectories, GNSS receiver may lose satellite signals when the car is in tunnels or canyons, cameras may be affected by sunlight or featureless environment, and LiDAR may suffer from insufficient data points (i.e., sparse point clouds) or reflective surfaces such as mirrors and windows. For this reason, the fusion of data from these sensors can overcome the problems presented above and can introduce stability and safety in the system. The information from these sensors is processed by an estimation algorithm to determine the variable of interest. The GNSS is used for localization, however, Claudine et al. (2021) states that GNSS positioning alone is unreliable because of interferences caused by tall trees, buildings, and tunnels. To avoid GNSS drawbacks, cameras and LiDARs are fused to help the localization process.

Cameras perceive the environment intuitively in a human-like manner. Although they are among the most commonly used sensors for localization and navigation, they do not always offer the maximum safety and efficiency. Compared to cameras, LiDARs are much more expensive sensors and offer more promising results, however, LiDARs can also fail on long-term trajectories. On this point, Zhang et al. (2015) has shown that better results can be obtained by fusing cameras and LiDARs for simultaneous localization and mapping (SLAM) algorithms. Before being fused, the sensors must first be calibrated. Calibration ensures that they work at the best possible scale and assure the best possible performance. The objective of calibration is to reduce measurement uncertainty and guarantee accuracy and consistency.

1.1 Autonomous driving relies on cameras and LiDARs

Research into autonomous driving is currently a hot topic. As Martins et al. (2020) stated, the autonomous driving field is no longer exclusive to academic research labs. To this end, development of all advanced driver assistance systems (ADAS) require an accurate perception, which means that information from multiple sensors need to be fused to perceive the environment. In this respect, Guindel et al. (2017) stated that 3D range scanners combined with stereo camera vision currently tend to be the

most utilized perception systems for autonomous cars.

Localization, mapping, and other traffic participant detection are essential tasks in autonomous driving. Approaches to solving these problems based only on inexpensive monocular and stereo camera sensors have resulted in drastically low accuracy, which according to Wang et al. (2019) is because of poor image-based depth estimation. Contrary to image-based depth estimation, LiDARs provide high-quality 3D point clouds, which are utilized to map the surrounding environment. Even though it is one of the most expensive components for a self-driving car, LiDAR remain indispensable in some situations where the vision-based solution might fail. Also, LiDAR prices are coming down to reasonable levels in the next few years, which will encourage autonomous driving companies to include them in their vehicles. To conclude, from an economical point of view, it is profitable to use vision-based solutions for autonomous vehicles, because camera sensors have lower costs. While from a safety and efficiency point of view it is better to use a combination of vision-based and ranging laser scans, in order to bring the highest accuracy and performance.

1.2 Research questions and goals

In this thesis, camera-LiDAR calibration methods are presented. The data set for the experiments was collected with an autonomous-capable Ford Mondeo Hybrid vehicle at the Finnish Geospatial Research Institute (FGI) in the autonomous research vehicle observatory (ARVO). Two iDS cameras are used to collect visual information and Velodyne VLS-128 LiDAR is used to measure point cloud data. We assume that the LiDAR is already intrinsically calibrated. The goal of the thesis is to implement and evaluate camera-LiDAR calibration methods which can improve the vehicle's perception system. For camera calibration, we aim to collect several data sets with and without the car windshield, in order to verify the windshield effect on calibration. We use several targets for data collection to verify if the type of calibration board brings any significant improvement on calibration. Camera-LiDAR extrinsic parameters are estimated based on 3D-2D (LiDAR-pixels) and 3D-3D (LiDAR-stereo camera) points in order to examine which method is more suitable. At the end of this work we will give answers to the following research questions:

1. What is the effect of the windshield on camera calibration?
2. Which is the best target for cameras and LIDAR calibration?
3. What is the best method for camera-LiDAR extrinsic calibration?

The goals are:

1. To perform mono and stereo camera calibration inside and outside the car.
2. To synchronize camera and LiDAR sensors.
3. To propose an automatic technique to extract point correspondences from the camera and LiDAR frames.

4. To estimate the extrinsic parameters for camera-LiDAR calibration.
5. To propose a method fo camera-LiDAR occlusion handling.
6. To fuse camera and LiDAR data.

1.3 Structure of the thesis

The thesis is organized as follows: Chapter 2 presents an overview of camera models and lens distortion, followed by mono and stereo calibration methods. LiDAR calibration is presented briefly, after which the camera-LiDAR extrinsic calibration is described. The platform used for the experiments is presented in Chapter 3. Also in that chapter, we present the research methods and materials. In Section 4 we present in detail and interpret the results obtained in mono and stereo calibration, as well as the extrinsic camera-LiDAR calibration. Chapter 5 presents a discussion on the obtained results. The conclusion are presented in Chapter 6.

2 Background

In this chapter, we present the mono and stereo camera calibration techniques. We discuss the distortion models and stereo vision based depth. The LiDAR is already internally calibrated, but for completeness, we briefly present LiDAR calibration techniques. In the last part, we present the camera-LiDAR extrinsic estimation methods.

2.1 Pinhole camera model

A camera sensor is a device that maps the 3D world points into a 2D image space Hartley et al. (2004). We define a 3D point $\mathbf{X} = (X, Y, Z)^\top$ and a 2D point $\mathbf{x} = (x, y)^\top$. The camera pinhole geometry is presented in Figure 1. The transformation

$$(X, Y, Z)^\top \rightarrow (f\frac{X}{Z}, f\frac{Y}{Z})^\top \tag{1}$$

is used to map the 3D point $(X, Y, Z)^\top$ into the image space, where f is the focal length. Note that in this case we assume that the focal lengths in x and y directions are the same. The center of the projection is called the *camera center*, point **C** in Figure 1. The *principal axis* is the line from *camera center* perpendicular to the *image plane*. The intersection of the *image plane* and *principal axis* is called *principal point*. Usually, *principal point* is the center or left top corner of the *image plane*. For

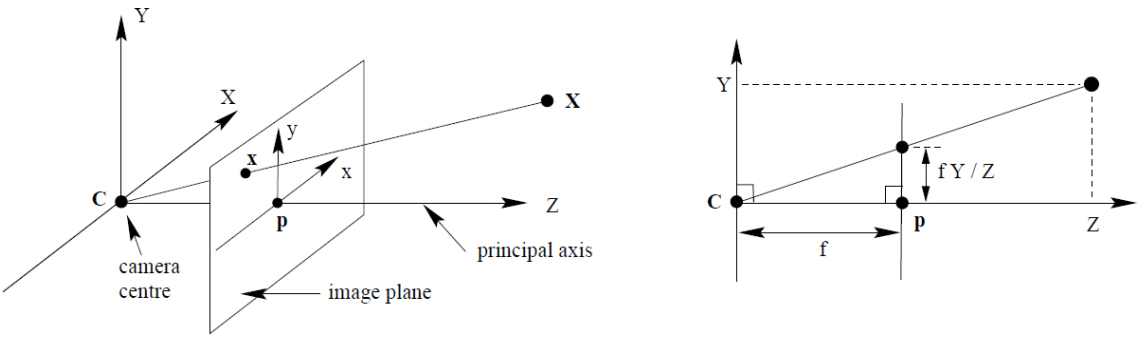

Figure 1: Camera pinhole model, left is the 2D case, on the right is the 1D case. Adapted from Hartley et al. (2004).

the sake of notation, the homogeneous vector representation is used to express the linear mapping

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

which can be rewritten in a compact form as

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \tag{3}$$

where $\mathbf{P}$ is the camera projection matrix as defined by Hartley et al. (2004), $\mathbf{x}$ is the 2D image point, and $\mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^\top$ is the homogeneous 3D point. In Equation (1), we assumed that the principal point lies on the origin coordinates of the image plane, in general, this is not true. The following transformation shows the general case when principal point and camera origin are not the same

$$(X, Y, Z)^\top \rightarrow (f\frac{X}{Z} + p_x, f\frac{Y}{Z} + p_y)^\top. \tag{4}$$

Similarly, we can rewrite Equation (2) as

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{5}$$

The internal calibration parameters are defined by matrix

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}, \tag{6}$$

where f is the focal length, and p_x and p_y define the optical center. Hartley et al. (2004) defines $\mathbf{K}$ as the *camera calibration matrix*. Similar definition of $\mathbf{K}$ can be found in Forsyth et al. (2012) and Szeliski (2010). Using the camera calibration matrix, Equation (3) becomes

$$\mathbf{x} = \underbrace{\mathbf{K}[\mathbf{I}|\mathbf{0}]}_{\mathbf{P}} \mathbf{X}_{cam}, \tag{7}$$

where $\mathbf{X}_{cam}$ is the 3D point in the camera coordinate system, $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is a block of 3×1 zeros. The transformation from world frame to camera frame is required to project an arbitrary 3D world point on image space. The mapping transformation consists of camera rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$. $\mathbf{R}$ and $\mathbf{t}$ are known as camera extrinsic parameters. Using camera extrinsic parameters, Equation (7) becomes

$$\mathbf{x} = \underbrace{\mathbf{K}[\mathbf{R}|\mathbf{t}]}_{\mathbf{P}} \mathbf{X}_{world}, \tag{8}$$

where $\mathbf{X}_{world}$ is a 3D point in the world frame, $\mathbf{K}$ is the camera intrinsic matrix, and $\mathbf{R}$ and $\mathbf{t}$ are camera extrinsic parameters with

$$[\mathbf{R}|\mathbf{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}. \tag{9}$$

Therefore, the projection matrix $\mathbf{P}$ is expressed as

$$\mathbf{P} = \underbrace{\begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{[\mathbf{R}|\mathbf{t}]} = \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{3\times4 \ \ matrix}. \tag{10}$$

2.2 Ground truth data for camera calibration

The ground truth point correspondences between the 3D world and 2D pixels space are required to estimate the values of the projection matrix **P**. Camera 2D points are estimated based on features in the image space. To obtain the 3D values of the points, we should accurately measure their coordinates relative to the camera frame. A simpler solution to collect 3D world points is to use a calibration plane/target with a known structure and dimensions of the features. As shown in Figure 2, the calibration board might be a simple chessboard, circle grid or any other template that can be detected in camera space.

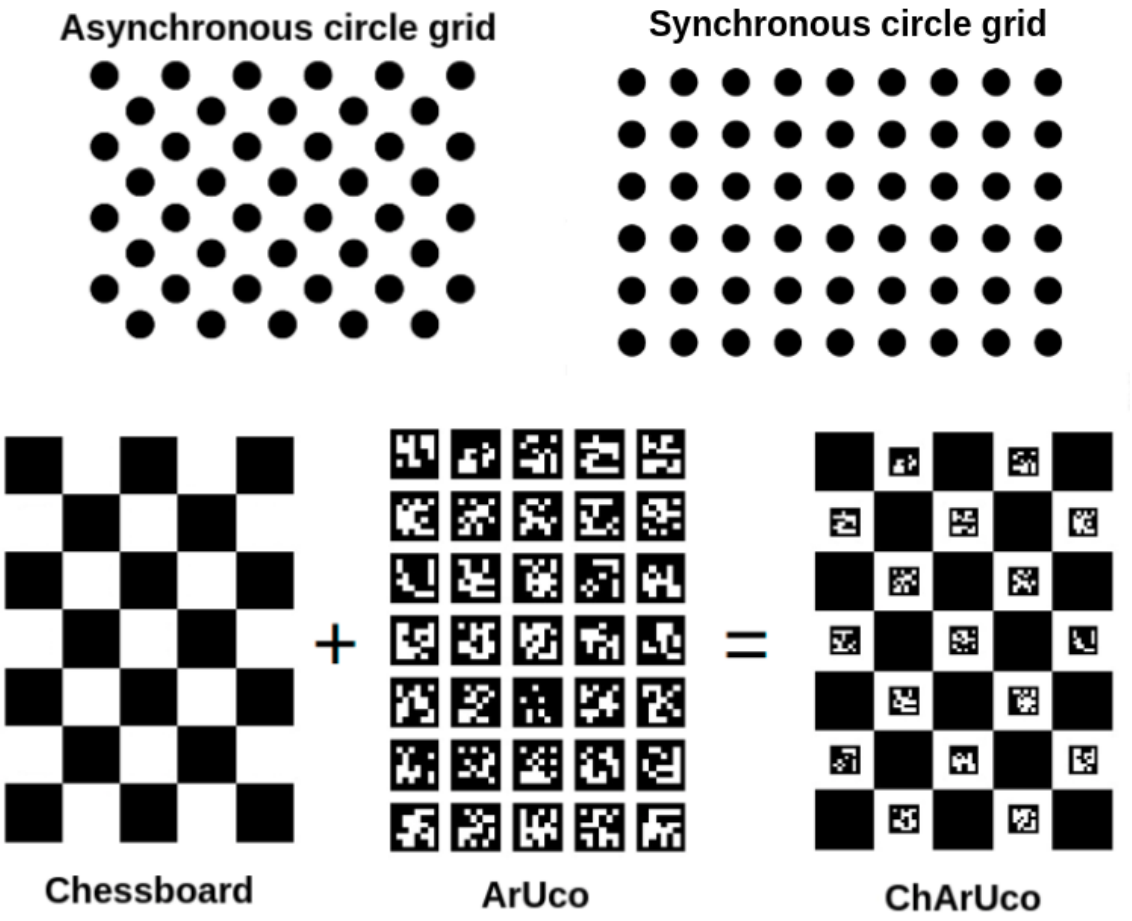

Figure 2: Camera calibration boards, circle grids, chessboard, ArUco, ChArUco - a combination of the previous two.

The most popular calibration pattern design is the chessboard. It can be detected by binarizing image and extracting black squares that meet the criteria imposed by the user. After detecting the board, each corner of the template is estimated with high accuracy, because the corners are unbiased under lens distortion or perspective transformations. In OpenCV chessboard detection module, the entire calibration board is required to be visible in the camera FOV and this is the main drawback,

as it is problematic to collect points that belong to the very edges of the frame and these points are most affected by lens distortion. Also, to be rotation invariant, the number of columns must be even and the number of rows must be odd, or vice-versa. If both are even or odd, there is 180-degree rotation ambiguity for stereo calibration.

Circle grid is another calibration target that can be detected based on the white or black circles on it. Circles are detected as blob objects. The regular structure is identified based on the detected circle points. In contrast to the corners of the chessboard, the circles can be seen as ellipses through the camera lens, which means that the detection of candidate circles is affected by the lens distortion.

ChArUco calibration target overcomes the limits of the chessboard and circle grid mentioned above. As the name suggests, it is a combination of a chessboard and ArUco markers presented by Marut et al. (2019). An ArUco marker is placed in each white cell on the chessboard and it can be uniquely identified. This eliminates the problem of rotation ambiguity. Also, there is no need for the entire calibration board to be visible in the camera FOV, this allows point collection from the very edges of the frames.

2.3 Camera lens and distortion parameters

In the pinhole camera model, we presented the tranformation of 3D points to 2D image space using projection matrix $\mathbf{P}$. However, the pinhole model is a mathematical ideal case, which cannot be directly applied to real cameras. For a general camera sensor, the smaller the aperture (hole) is, the sharper and clear the resulting images are. However, this limits the number of photons that hit the image sensor and the images are dark. If the aperture is increased, multiple rays from the 3D world are incident on the same part of the sensor screen, which makes the image blurred. We are interested in getting a sharp image, and at the same time to get more light rays to make the image brighter. As shown in the following figure, replacing the pinhole with the lens allows us to get a sharp and bright image by capturing multiple light rays with lens. However, lenses introduce distortion problem.

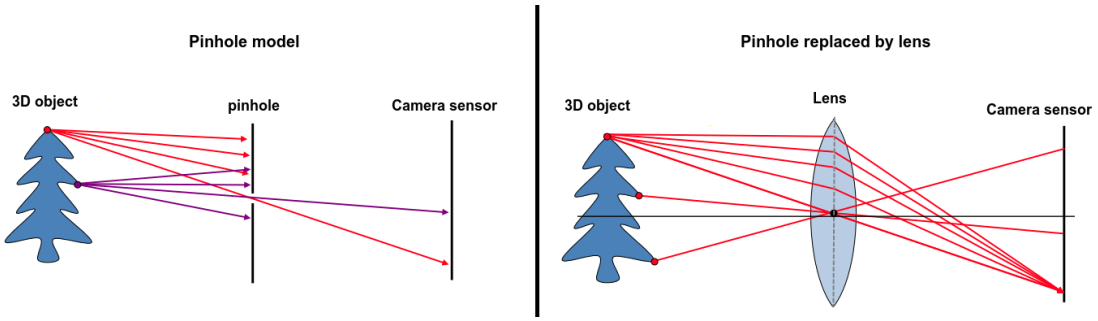

Figure 3: Camera pinhole replaced by lens. Replacing the pinhole with a lens, allows gathering multiple light rays that makes the resulted image brighter. Adapted from Scaramuzza et al. (2011).

On this point, Wang et al. (2008) states that there are two principal components of distortion shown in Figure 4.

1. **Radial distortion** - caused by the variations of the light refractions, on camera lens, also called *barrel distortion*.

2. **Tangential distortion** - a problem in the manufacturing of cameras, caused by a small displacement of the lens center from the optical axis.

The standard distortion model includes 5 parameters, (k_1, k_2, k_3 for radial distortion and p_1, p_2 for tangential distortion). Besides the standard model, Martins et al. (2020) present rational, thin prism and tilted distortion models.

- **Standard distortion** - a model with three parameters (k_1, k_2, k_3) for radial

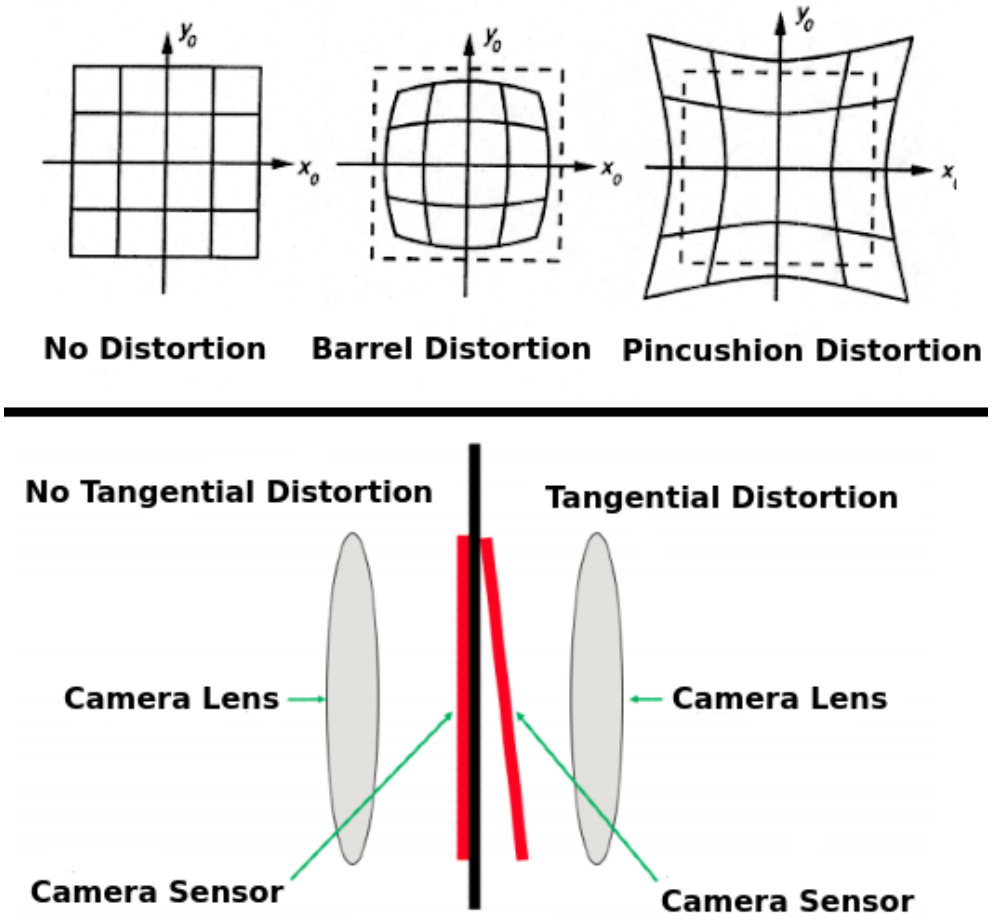

Figure 4: The standard camera lens distortion model. Top - radial distortion, caused by light refraction. Bottom - tangential distortion, caused by inaccuracies in manufacturing of cameras. Adapted from Chiou (2017).

distortion and two parameters (p_1, p_2) for tangential distortion, this is known as Duane (1971) model.

- **Rational distortion** - besides the standard distortion model parameters, includes extra three radial distortion parameters, (k_4, k_5, k_6), which gives better results with higher distortions.
- **Thin prism distortion** - this model includes the standard distortion and extra four parameters, (s_1, s_2, s_3, s_4). Wang et al. (2008) argues that Thin prism distortion is caused by slight tilt of lens or image sensor array which can cause additional radial and tangential distortion.
- **Tilted distortion** - the lens tilting around x, y axes is represented by two extra parameters (τ_x, τ_y).

As reported by Martins et al. (2020) the projected pixel is affected according to the following equation

$$\begin{aligned} x &= x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2(r^2 + 2x'^2) + s_1 r^2 + s_2 r^2, \\ y &= y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_2 x' y' + p_1(r^2 + 2y'^2) + s_3 r^2 + s_4 r^2, \end{aligned} \tag{11}$$

where (x, y) stands for corrected pixel, (x', y') is the projected pixel, and r is the radial distance of the projected point (x', y') with $r^2 = x'^2 + y'^2$. The effect of changing the distortion parameters is presented in Figure 5. The importance of distortion

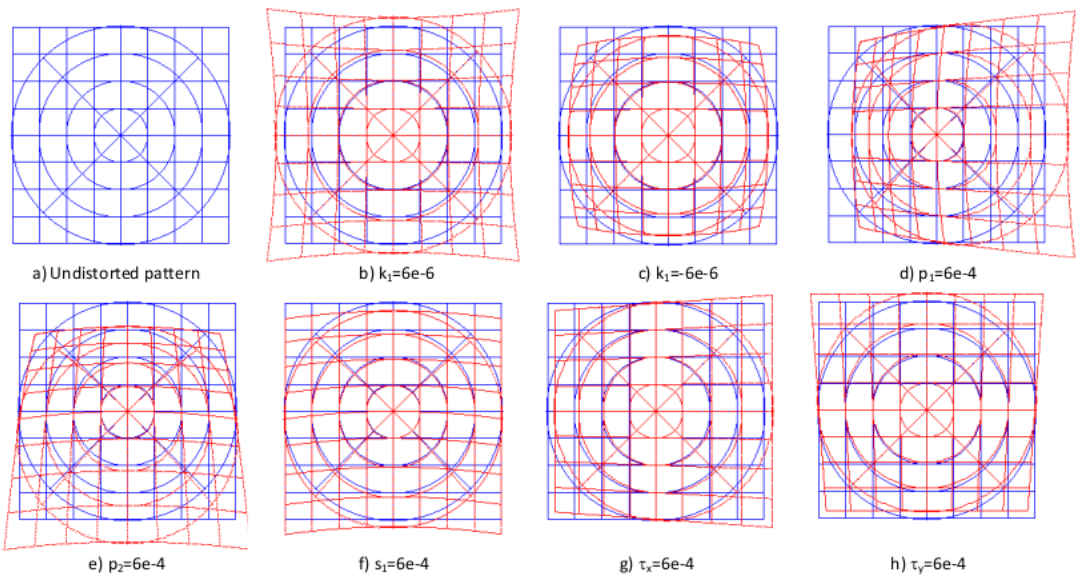

Figure 5: The effect of the distortion parameters on the image frame. Adapted from Martins et al. (2020).

parameter estimation is clear, these parameters are used to correct the image, thus improve the robustness and accuracy of most computer vision algorithms. Let $\mathbf{x}'$ be projected sensor point and $\mathbf{x}$ the observed sensor point, thus, the observed distortion vector is

$$\hat{\mathbf{d}} = \mathbf{x} - \mathbf{x}'. \tag{12}$$

The resulting vector after applying distortion model from Equation (11) on projected point $\mathbf{x}'$ is referred as model distortion vector $\mathbf{d}$. As Burger (2016) described, the distortion parameters can be estimated using least-squares fitting, minimizing the difference between the model distortion and observed distortion as

$$\underset{k_1,k_2,\dots}{\arg\min} \sum_{i=1,j=1}^{N,M} ||\mathbf{d}_{i,j} - \hat{\mathbf{d}}_{i,j}||^2, \tag{13}$$

where N and M are the 2D image width and height.

2.4 Camera calibration methods

In this section we present the intrinsic ($\mathbf{K}$ matrix and distortion) and extrinsic (rotation $\mathbf{R}$ and translation $\mathbf{t}$) camera parameter estimation.

2.4.1 Direct linear transformation

Direct linear transformation (DLT) is a method to estimate the projection matrix $\mathbf{P}$ from given point correspondences (Shapiro 1978). Equation (7) can be rewritten as

$$\lambda \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{x}} = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{23} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\mathbf{X}}, \tag{14}$$

where λ is a scaling factor, $(x, y, 1)^\top$ is camera pixel and $(X, Y, Z, 1)^\top$ is a 3D world point in camera frame written in a homogeneous form. Equation (14) can be rewritten as a system with three equations

$$\begin{cases} \lambda x &= Xp_{11} + Yp_{12} + Zp_{13} + p_{14} \\ \lambda y &= Xp_{21} + Yp_{22} + Zp_{23} + p_{24} \\ \lambda &= Xp_{31} + Yp_{32} + Zp_{33} + p_{34} \end{cases}$$

We can substitute λ from the last equation into first two, which gives

$$\begin{cases} X(p_{11} - p_{31}x) + Y(p_{21} - p_{32}x) + Z(p_{13} - p_{33}x) + (p_{14} - p_{34}x) &= 0 \\ X(p_{21} - p_{31}y) + Y(p_{22} - p_{32}y) + Z(p_{23} - p_{33}y) + (p_{24} - p_{34}y) &= 0 \end{cases}$$

As described by Dubrofsky (2009), for multiple points we can stack the equations and rewritten the above system as a matrix multiplication

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -Z_1x_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_1 & -Y_1y_1 & -Z_1y_1 & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -X_ix_i & -Y_ix_i & -Z_ix_i & -x_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -X_iy_i & -Y_iy_i & -Z_iy_i & -y_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -Z_nx_n & -x_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -X_ny_n & -Y_ny_n & -Z_ny_n & -y_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

or in a compact form

$$\mathbf{Ap} = \mathbf{0} \tag{15}$$

with $\mathbf{A}$ being a $2n \times 12$ matrix and $\mathbf{p}$ unknown projection matrix written in a vector form. The vector $\mathbf{p}$ has 12 unknown components, but only 11 DOF because the last unknown component is the scale. Since each pair of point correspondence gives two equations, at least six points are required to estimate the solution. Solution of Equation (15) can be computed via SVD($\mathbf{A}$) = $\mathbf{U\Sigma V}^\top$, the unknown vector $\mathbf{p}$ is the last column of $\mathbf{V}$ matrix. As Hartley et al. (2004) described, the intrinsic and extrinsic parameters can be extracted from projection matrix $\mathbf{P}$ via RQ matrix decomposition (the QR decomposition with reversed order).

The standard DLT algorithm can be used under the assumption that the pixels and world point correspondences are known. Sometimes, in real environments, it is hard to gather exact 3D positions of the points. A simplified method is used in the Zhang (2000) technique, where points are collected using calibration targets presented in Section 2.2, under the assumption that z coordinate of each 3D point is zero.

2.4.2 Plane-based calibration

According to Burger (2016) the mono camera calibration is done following the next steps:

1. Collect images using a target with a known structure. The points are collected either by moving the camera around the target or moving the target in front of the camera. For every image, 2D pixels and their corresponding corners are extracted from the target.

2. The 3D homography is estimated for each view, using a modified DLT algorithm. In the standard DLT algorithm the mapping is defined as

$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_{world}, \tag{16}$$

where

$$\mathbf{K}[\mathbf{R}|\mathbf{t}] = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{23} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}, \tag{17}$$

which means that parameters are estimated all together from projection matrix $\mathbf{P}$. Since the 3D points have the z axis zero, the projection takes the form of

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ r_0 & r_1 & r_2 & t \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \xrightarrow[\mathrm{Z}=0]{\text{Since}} \underbrace{\mathbf{K} \begin{bmatrix} \vdots & \vdots & \vdots \\ r_0 & r_1 & t \\ \vdots & \vdots & \vdots \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{18}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{19}$$

where $\mathbf{H}$ is a 3×3 transformation matrix. Similar to DLT, for multiple points, the equations are stacked and expressed as a homogeneous linear system

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1y_1 & -Y_1y_1 & -y_1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_i & Y_i & 1 & 0 & 0 & 0 & -X_ix_i & -Y_ix_i & -x_i \\ 0 & 0 & 0 & X_i & Y_i & 1 & -X_iy_i & -Y_iy_i & -y_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & 1 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -x_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -X_ny_n & -Y_ny_n & -y_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{20}$$

The initial guess of $\mathbf{H}$ is computed using SVD after which the solution can be refined by minimizing the following error

$$\arg\min_{\mathbf{H}} \sum_{j=0}^{N-1} \left\| \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} - \mathbf{H} \begin{bmatrix} X_j \\ Y_j \\ 1 \end{bmatrix} \right\|^2, \tag{21}$$

where N is the number of point correspondences. Any optimization technique can be used to minimize the above error, OpenCV uses Levenberg-Marquardt (Lourakis 2005) optimization with initial guess provided by SVD. The value function used for minimization is defined as

$$\mathrm{f}(\mathbf{X}, \mathbf{H}) = \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{22}$$

and the Jacobian is defined as

$$\frac{\partial \mathrm{f}}{\partial(\mathbf{H})} = \begin{bmatrix} \frac{\partial x_1}{\partial h_1} & \frac{\partial x_1}{\partial h_2} & \frac{\partial x_1}{\partial h_3} & \frac{\partial x_1}{\partial h_4} & \frac{\partial x_1}{\partial h_5} & \frac{\partial x_1}{\partial h_6} & \frac{\partial x_1}{\partial h_7} & \frac{\partial x_1}{\partial h_8} & \frac{\partial x_1}{\partial h_9} \\ \frac{\partial y_1}{\partial h1} & \frac{\partial y_1}{\partial h_2} & \frac{\partial y_1}{\partial h_3} & \frac{\partial y_1}{\partial h_4} & \frac{\partial y_1}{\partial h_5} & \frac{\partial y_1}{\partial h_6} & \frac{\partial y_1}{\partial h_7} & \frac{\partial y_1}{\partial h_8} & \frac{\partial y_1}{\partial h_9} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_n}{\partial h_1} & \frac{\partial x_n}{\partial h_2} & \frac{\partial x_n}{\partial h_3} & \frac{\partial x_n}{\partial h_4} & \frac{\partial x_n}{\partial h_5} & \frac{\partial x_n}{\partial h_6} & \frac{\partial x_n}{\partial h_7} & \frac{\partial x_n}{\partial h_8} & \frac{\partial x_n}{\partial h_9} \\ \frac{\partial y_n}{\partial h_1} & \frac{\partial y_n}{\partial h_2} & \frac{\partial y_n}{\partial h_3} & \frac{\partial y_n}{\partial h_4} & \frac{\partial y_n}{\partial h_5} & \frac{\partial y_n}{\partial h_6} & \frac{\partial y_n}{\partial h_7} & \frac{\partial y_n}{\partial h_8} & \frac{\partial y_n}{\partial h_9} \end{bmatrix}. \tag{23}$$

3. Similar to DLT, the computed homography is decomposed into intrinsic and extrinsic parameters. The camera center or translation of the camera in world coordinates $\mathbf{C} = (X, Y, Z, W)^\top$ is a point where $\mathbf{PC} = \mathbf{0}$ or

$$\mathbf{RC} + \mathbf{t} = \mathbf{0} \quad \Rightarrow \quad \mathbf{C} = -\mathbf{R^{-1}t} \quad \Rightarrow \quad \mathbf{C} = -\mathbf{R}^\top\mathbf{t}, \tag{24}$$

where $\mathbf{t}$ is the relative translation, $\mathbf{R}$ is relative rotation between camera and world frame, and

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] = \mathbf{K}[\mathbf{R}| - \mathbf{RC}] = [\mathbf{KR}| - \mathbf{KRC}]. \tag{25}$$

We have $\mathbf{R^{-1}} = \mathbf{R}^\top$ since $\mathbf{R}$ is a rotation matrix and it is orthonormal. The homography matrix $\mathbf{H}$ takes the role of $\mathbf{P}$ and is decomposed into intrinsic and rotation matrix.

4. Since the approximate values of the intrinsic parameters are known, techniques presented in Section 2.3 are applied to estimate the lens distortion parameters.

5. The last step of the plane based calibration is to combine all solutions together. Intrinsic parameters ($\mathbf{K}$ matrix) are the same for all views, while rotation and translation parameters are estimated for each view independently. The total projection error is defined as

$$\arg\min_{\mathbf{K},\mathbf{R}_j,\mathbf{t}_j} = \sum_{i=0}^{N} ||\mathbf{x}_i - \mathbf{x}'_i||^2, \tag{26}$$

where $\mathbf{x}$ is the observed pixel, $\mathbf{x}'$ is the projected pixel, and N is the total number of points in image j. Using $\mathbf{P}$ matrix the above equation becomes

$$\arg\min_{\mathbf{K},\mathbf{R}_j,\mathbf{t}_j} \sum_{i=0}^{N} ||\mathbf{x}_i - \mathbf{P}\mathbf{X}_i||^2 = \arg\min_{\mathbf{K},\mathbf{R}_j,\mathbf{t}_j} \sum_{i=0}^{N} ||\mathbf{x}_i - \mathbf{K}[\mathbf{R}_j|\mathbf{t}_j]\mathbf{X}_i||^2, \tag{27}$$

where j is the image view.

The plane-based calibration steps are covered in OpenCV *calibrateCamera* function (Bradski 2000).

2.5 Stereo vision

Stereo vision (Ayache 1991) is the principle of using two cameras to measure the depth for each pixel. The geometry is motivated by searching for pixel correspondences in stereo images. Besides the intrinsic calibration, in stereo case we have the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between cameras. The stereo setup is presented in Figure 6. The translation between cameras is called *baseline*. The camera origins ($\mathbf{O}$ and $\mathbf{O}'$) and 3D point $\mathbf{X}$ are coplanar and form the epipolar plane (gray area in Figure 6). The intersection of the image planes with the baseline are called *epipoles*. The intersection of the epipolar plane with the image planes form *epipolar lines*.

In the epipolar constraint, for each observed point $\mathbf{x}$ in the left image, we search for the corresponding point $\mathbf{x}'$ in the right image. As shown in Figure 6, the left camera coordinate system is a world frame and the rotation for the left camera is $\mathbb{I}$ and translation is zero. The right camera is transformed relative to left camera. Potential matches for point $\mathbf{x}$, have to lie on the corresponding epipolar line *l'*, and vice-versa, potential matches for $\mathbf{x}'$, lie on the epipolar line *l*. If the intrinsic and extrinsic parameters of the cameras are known, the projection matrices are given by:

$$\begin{aligned} \mathbf{P}_{left} &= \mathbf{K}_{left}\ [\,\mathbf{I}\,|\mathbf{0}], \\ \mathbf{P}_{right} &= \mathbf{K}_{right}[\mathbf{R}|\mathbf{t}]. \end{aligned} \tag{28}$$

The cross product of every vector by itself is zero, $\vec{\mathbf{a}} \times \vec{\mathbf{a}} = 0$, and any cross product

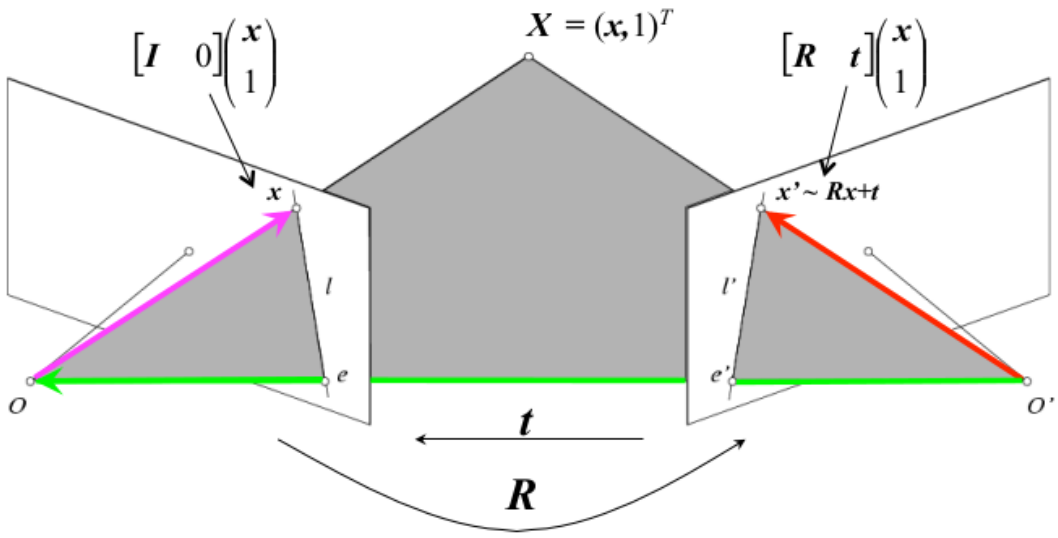

Figure 6: Epipolar geometry. $\mathbf{O}$-left camera origin, $\mathbf{O}'$-right camera origin, $\mathbf{X}$-3D world point, $\mathbf{x}$-pixel projection of $\mathbf{X}$ point on left camera frame, $\mathbf{x}'$-pixel projection of $\mathbf{X}$ point on right camera frame. Adapted from Hartley et al. (2004).

can be written in a matrix form

$$\overrightarrow{\mathbf{a}} \times \overrightarrow{\mathbf{b}} = [\mathbf{a}]_{\times} \overrightarrow{\mathbf{b}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{29}$$

Vectors $\mathbf{Rx}$, $\mathbf{t}$ and $\mathbf{x}'$ from Figure 6 are coplanar, *triple dot product rule* (Szabo 2015) can be applied, which results in

$$\underbrace{\mathbf{x}'[\mathbf{t} \times (\mathbf{Rx})] = 0}_{\textit{triple} \quad \textit{dot} \quad \textit{product}} \xrightarrow[\text{Eq (29)}]{\text{Using}} \mathbf{x}'^{\top}[\mathbf{t}]_{\times}\mathbf{Rx} = 0 \rightarrow \mathbf{x}'^{\top}\mathbf{Ex} = 0, \tag{30}$$

where $[\mathbf{t}]_{\times}$ is the translation vector written in a matrix form and $\mathbf{E}$ is the *essential matrix* (Yang et al. 2014). In this case $\mathbf{Ex}$ is the epipolar line associated with $\mathbf{x}$ point ($l' = \mathbf{Ex}$) and $\mathbf{E}^{\top}\mathbf{x}'$ is the epipolar line for $\mathbf{x}'$. If $\mathbf{E}$ matrix is known, we can extract rotation and translation between cameras from it, as presented by Horn (1990). Usually, $\mathbf{E}$ matrix is computed when the cameras are calibrated (the intrinsic parameters are known). If $\mathbf{K}_{left}$ and $\mathbf{K}_{right}$ are unknown, we can rewrite the result of Equation (30) in terms of unknown intrinsic parameters

$$\hat{\mathbf{x}}'^{\top}\mathbf{E}\hat{\mathbf{x}} = 0, \tag{31}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are points on the left and right camera, written in terms of unknown intrinsic matrices $\mathbf{K}_{left}$ and $\mathbf{K}_{right}$. Luong et al. (1996) presented a method of computing rotation and translation between stereo cameras, with unknown internal calibration where essential matrix is replaced

$$\hat{\mathbf{x}}'^{\top}\mathbf{E}\hat{\mathbf{x}} = 0 \rightarrow \mathbf{x}'^{\top}\mathbf{Fx} = 0, \tag{32}$$

where $\mathbf{F}$ is the *fundamental matrix* and

$$\mathbf{F} = \mathbf{K}_{right}^{-\top}\mathbf{E}\mathbf{K}_{left}^{-1}. \tag{33}$$

Stereo data points can be collected in a similar manner as for mono camera calibration with the constraint that the same point should be visible in both cameras at the same time. Any patterns presented in Section 2.2 can be used to collect point correspondences for the left and right camera. For now, let's assume the point correspondences are known $\mathbf{x} = (u, v, 1)^{\top}$ and $\mathbf{x}' = (u', v', 1)$. *Eight-point algorithm* (Hartley 1997 and Chojnacki et al. 2003) can be used to estimate the $\mathbf{F}$ matrix.

2.5.1 The eight-point algorithm

The eight-point algorithm (Hartley 1997) is used to estimate the values of $\mathbf{F}$ matrix, given the point correspondences on the left and right camera. It describes the

relationship between stereo images as

$$\underbrace{\begin{bmatrix} u' & v' & 1 \end{bmatrix}}_{\mathbf{x}'} \underbrace{\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}}_{\mathbf{F}-matrix} \underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\mathbf{x}} \rightarrow \underbrace{\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}}_{\mathbf{f}} = 0, \tag{34}$$

or in a compact form

$$\mathbf{Af} = 0. \tag{35}$$

The homogeneous linear system is solved using SVD decomposition, and the solution is the eigenvector that corresponds to the smallest singular eigenvalue. With known $\mathbf{F}, \mathbf{K}_{left}$, and $\mathbf{K}_{right}$ matrices, we can infer $\mathbf{E}$ from Equation (33). According to Luong (1993), $\mathbf{E}$ is decomposed in rotation $\mathbf{R}$ and translation $\mathbf{t}$ in the following manner:

- Take $\mathbf{U\Sigma V}^\top = \text{SVD}(\mathbf{E})$ where $\mathbf{U}$ and $\mathbf{V}$ are 3×3 orthogonal matrices and $\mathbf{\Sigma}$ is 3×3 diagonal matrix.
- Define $\mathbf{W}$ matrix as:
$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$
where $\mathbf{W}^{-1} = \mathbf{W}^\top$.
- Define $\mathbf{R}$ and $[\mathbf{t}]_\times$ as:
$$\mathbf{R} = \mathbf{UW}^{-\mathbf{1}}\mathbf{V}^\top,$$
$$[\mathbf{t}]_\times = \mathbf{UW\Sigma U}^\top,$$
where $\mathbf{R}$ is rotation and $[\mathbf{t}]_\times$ is the translation vector written in a matrix form.

2.5.2 Stereo rectification and correspondence search

Point depth can be inferred from pixel correspondences between images (Zou et al. 2010). For each pixel in the left image, we search the best matching pixel in the right image. Both images should be aligned, such that the motion is horizontal. By rectifying the images, we ensure that epipolar lines are horizontal and the pixel correspondence from the left image lies on the same row in the right image. This reduces the correspondence search space to 1D after rectification. Stereo rectification process is illustrated in Figure 7. According to Fusiello et al. (2000) the main steps to rectify images are:

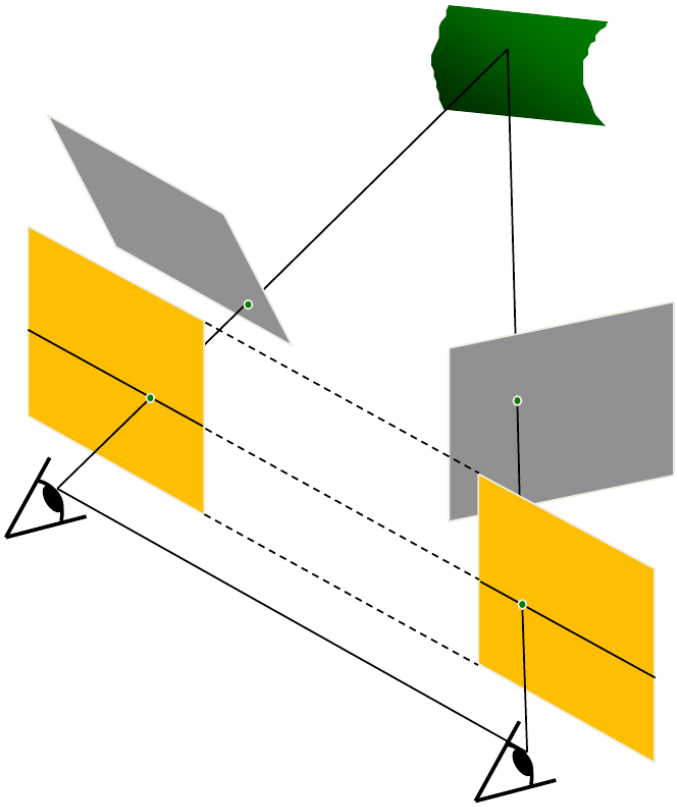

Figure 7: Stereo rectification, the gray planes are the original left and right image, the yellow planes are the stereo-rectified images. Adapted from Loop et al. (1999).

1. Rotate left camera such that its image plane is parallel to the baseline.
2. Apply the same rotation to right camera.
3. Adjust the scale in both cameras.

Searching the point correspondences between images is a challenging problem. The standard correlation-based methods compute the common regions for all pixels in the left and right image (Hall 1999). Figure 8 presents the block matching technique and disparity computation (Georgoulas et al. 2008). For each block from the left image, we search the correspondence on the epipolar line, in the right image. The blocks are matched using the sum of squared differences (SSD) cost function

$$\mathrm{SSD}(d) = \sum_{x,y \in \mathbf{W_m}}^{M} [\mathbf{I}_L(x,y) - \mathbf{I}_R(x+d,y)]^2 \quad = \quad ||\mathbf{W}_L - \mathbf{W}_R(d)||^2, \tag{36}$$

where $\mathbf{I}_L$ and $\mathbf{I}_R$ are left and right image, $\mathbf{W}_L$ and $\mathbf{W}_R$ are the left and right windows, M is the number of windows and d is pixel disparity. As shown in the bottom of the Figure 8, the matched block from the right image correspond to the lowest SSD cost

$$d^* = \arg\min_d ||\mathbf{W}_L - \mathbf{W}_R(d)||^2. \tag{37}$$

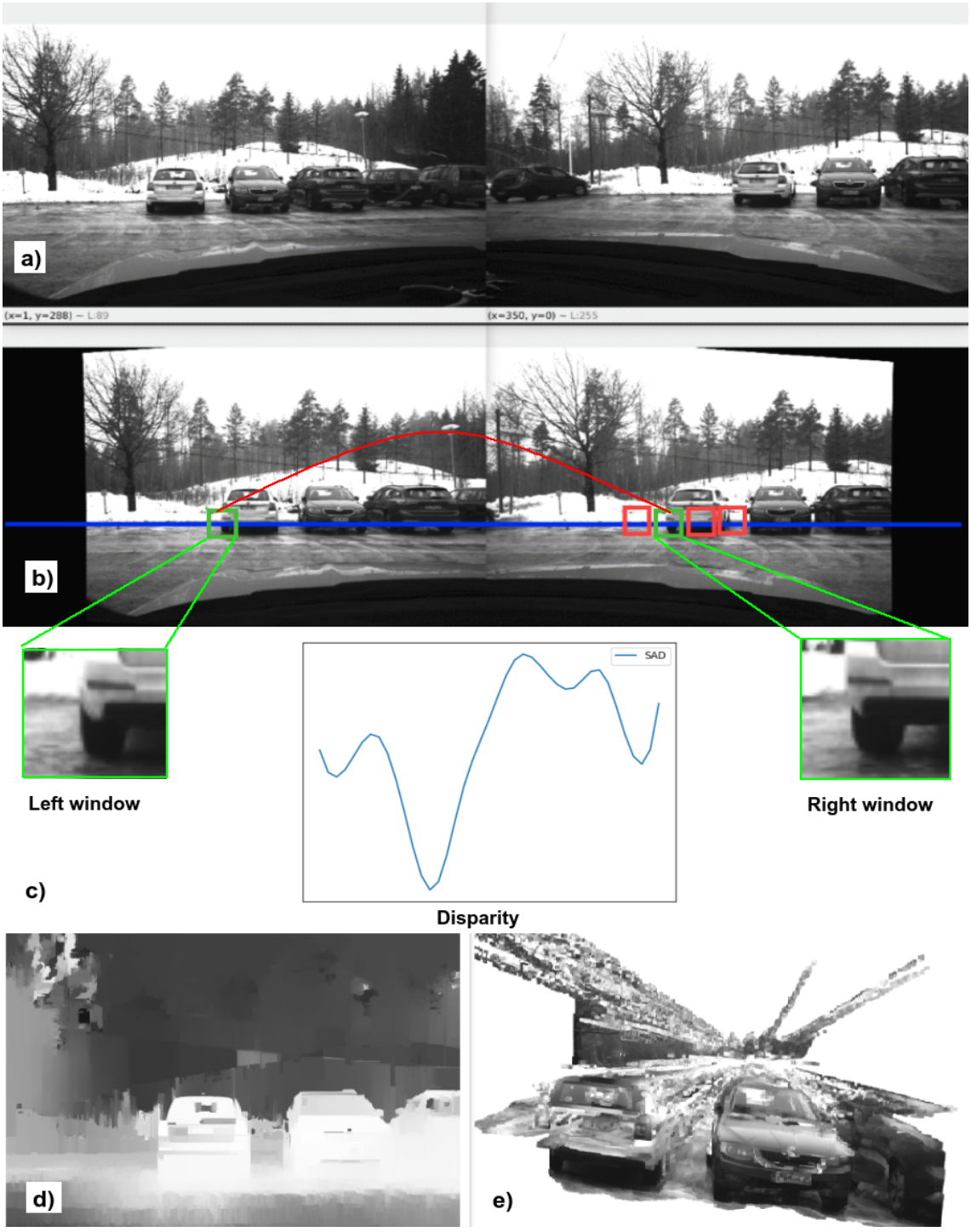

Figure 8: Demonstration of stereo rectification and correspondence via correlation. a) original left and right image, b) rectified images, the blue line is the epipolar line after the rectification process. c) blocks from left and right image, and their matching cost d) disparity map, e) reconstructed point cloud.

SSD cost function penalizes large errors more than small ones. The sum of absolute difference (SAD) cost function

$$\mathrm{SAD}(d) = \sum_{x,y \in \mathbf{W}_m}^{M} |\mathbf{I}_L(x,y) - \mathbf{I}_R(x+d,y)| \tag{38}$$

equally penalizes the errors. Until now we have assumed that pixel intensity in the left and right image is the same, however, in some cases, blocks may have different illuminations. Normalized cross-correlation (NCC) cost function

$$\mathrm{NCC}(d) = \sum_{x,y \in \mathbf{W}_m}^{M} \hat{\mathbf{I}}_L(x,y)\hat{\mathbf{I}}_R(x+d,y), \tag{39}$$

overcomes this issue, it is less sensitive to changes in illumination and therefore can result in a more accurate disparity map. The normalized pixel $\hat{\mathbf{I}}(x,y)$ is computed as

$$\hat{\mathbf{I}}(x,y) = \frac{\mathbf{I}(x,y) - \mathbf{I}_{mean}}{||\mathbf{I} - \mathbf{I}_{mean}||}. \tag{40}$$

According to Hirschmuller et al. (2007), NCC blurs the regions of discontinuity more than other matching cost methods. This happens due to large errors caused by outliers in NCC cost computation. A demonstration of the block matching is shown in Figure 8 c). We notice that disparity gets smaller with increasing depth. The objects closer to the camera have brighter disparity regions, while further regions are darker. The block size parameter influences the accuracy of the disparity since it enforces the block region to have the same depth, therefore, it should be tuned accordingly. The effect of window size is presented in Figure 9. Too big block size has a smooth effect on the disparity map, however, there is a loss of data, since the algorithm enforces all pixels within the same block to have the same depth. On the

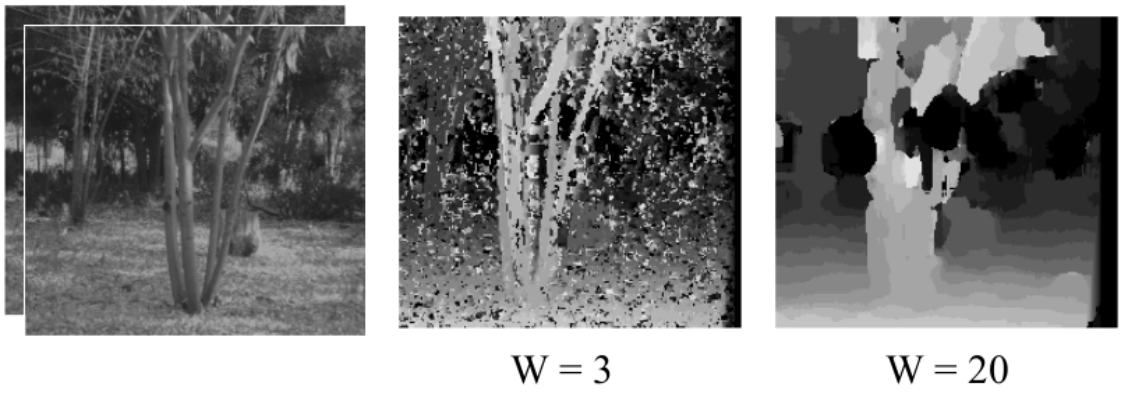

Figure 9: Effect of window size on disparity computation. Small window size results in more details and more noise. Big window size results in smooth disparity map, but less details. Adapted from Loop et al. (1999).

other hand, smaller block size can capture more pixels depth information, but the resulted disparity map is noisy.

The block matching method for disparity computation has some disadvantages: it suffers from textureless surfaces (i.e., not enough features to detect) and therefore cannot estimate the depth for those regions. Also, occlusions and repetition might produce ambiguities. In comparison to correlation-based, feature-based methods search for the sparse set of correspondences, typically most similar feature pairs like corners, edges, and lines. There are different techniques to compute the disparity, for example Wang et al. (2019) use a convolutional neural network to predict the disparity map. Godard et al. (2017) and Mac et al. (2019) have shown promising results in unsupervised monocular depth estimation, however, the results are not yet as accurate as LiDAR measurements.

2.5.3 Depth from disparity

Given stereo calibrated cameras, we can fuse them to compute depth information by using disparity (Georgoulas et al. 2008). Disparity means a lack of similarity or the difference, it is the difference between pixel points on the left and right camera. From similar triangles in Figure 10 we have

$$\frac{x}{f} = \frac{B_1}{z}, \tag{41}$$

$$\frac{-x'}{f} = \frac{B_2}{z}. \tag{42}$$

By summing Equations (41) and (42) we get

$$\frac{x - x'}{f} = \frac{B_1 + B_2}{z}, \tag{43}$$

$$disparity = x - x' = \frac{Bf}{z}, \tag{44}$$

where f is the focal length, z is the depth, and B is the baseline. We can infer depth from disparity using Equation (44). With depth information, each pixel is re-projected on 3D world using

$$\begin{aligned} Z &= \frac{Bf}{x - x'} \quad , \\ X &= \frac{Z(x - p_x)}{f} \quad , \\ Y &= \frac{Z(y - p_y)}{f} \quad , \end{aligned} \tag{45}$$

where p_x and p_y is the optical center and (X, Y, Z) is a 3D point with color from the image at pixel location (x, y). Figure 8 e) shows the reconstructed 3D point cloud based on stereo matching disparity map. The resulted 3D point cloud is dense and at the same time is noisy due to inaccuracies in the disparity map.

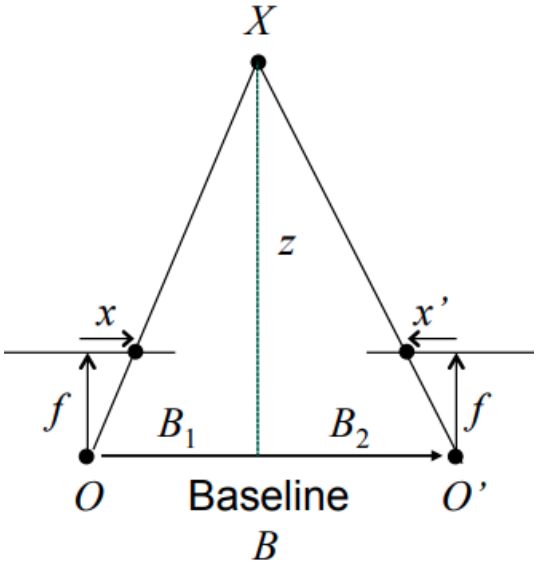

Figure 10: The stereo camera setup for disparity computation. Knowing the focal length and the baseline, the depth can be inferred using the difference between the projections of **X** point on each camera. Adapted from Hartley et al. (2004).

2.6 LiDAR calibration

LiDAR, also known as laser scanner (Lichti et al. 2000), estimates the distance to its surroundings by emitting laser light and measuring the time required for the light to return to the sensor. Molebny et al. (2010) state that origins of laser scanning comes from military research. Currently, LiDAR sensors are commonly used in autonomous vehicles. Hecht (2018) claims that the development of autonomous cars has led to an increase in demand for more accurate modelling of the surrounding environment, therefore, research in this area is a hot topic. LiDARs can be classified as single-beam (produces 2D data) or multi-beam (produces 3D data). In this thesis, we have used a multi-beam Velodyne (2018) LiDAR, which is provided with internal calibration parameters, but for completeness, we will describe the general steps for LiDAR calibration. As presented in Figure 11, a LiDAR sensor provides measurements in a polar coordinate system. The records consist of range R, azimuth angle α (which determines the direction of the measurement if we use a single-beam sensor), and the elevation angle ω. A point cloud can be generated by transforming the measurements into cartesian coordinate system. The transformation is presented in the following:

$$\begin{aligned} X &= R \cdot \cos(\omega) \cdot \sin(\alpha), \\ Y &= R \cdot \cos(\omega) \cdot \cos(\alpha), \\ Z &= R \cdot \sin(\omega). \end{aligned} \tag{46}$$

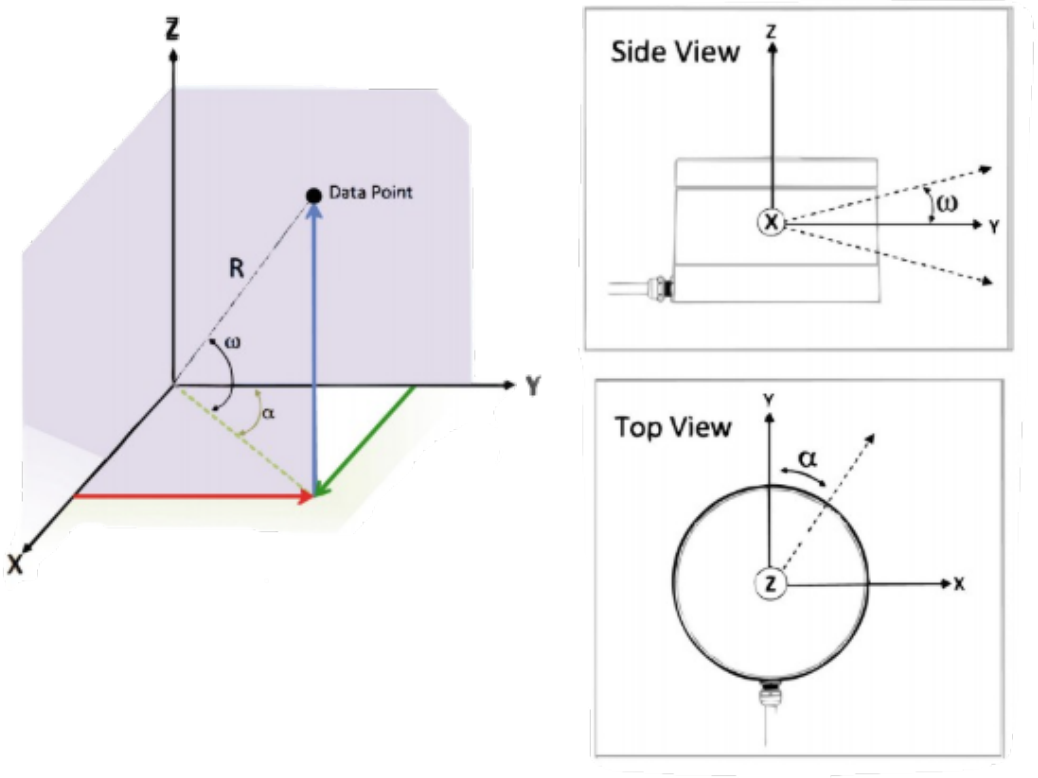

Figure 11: Raw sensor data conversion to (X, Y, Z) coordinate system. Adapted from Miądlicki et al. (2019).

The generated 3D point cloud is in the sensor coordinate frame. The radius R is computed using

$$\text{distance} = R = \frac{\text{acqusition time} - \text{shooting time}}{2} \cdot \text{speed of light}. \tag{47}$$

The time difference must be divided by two because the light ray travels the distance twice. As shown in Figure 12, Velodyne LiDAR has 360° horizontal FOV, but the vertical FOV and angular resolution depend on the number of LIDAR beams, for example, Velodyne *VLS-128* has $+15°$ to $-25°$ vertical FOV while Velodyne VLP-16 has only $+15°$ to $-15°$ vertical FOV (Velodyne 2018).

Similarly to the camera, LiDAR calibration parameters are divided into intrinsic and extrinsic parameters. LiDAR extrinsic parameters (rotation $\mathbf{R}$ and translation $\mathbf{t}$ between sensor and world frame) are defined as

$$\Theta_{extrinsic} = (\psi, \theta, \gamma, t_x, t_y, t_z), \tag{48}$$

were (ψ, θ, γ) are rotation parameters and (t_x, t_y, t_z) correspond to translation. To estimate the intrinsic parameters, each laser beam is inspected individually. For each

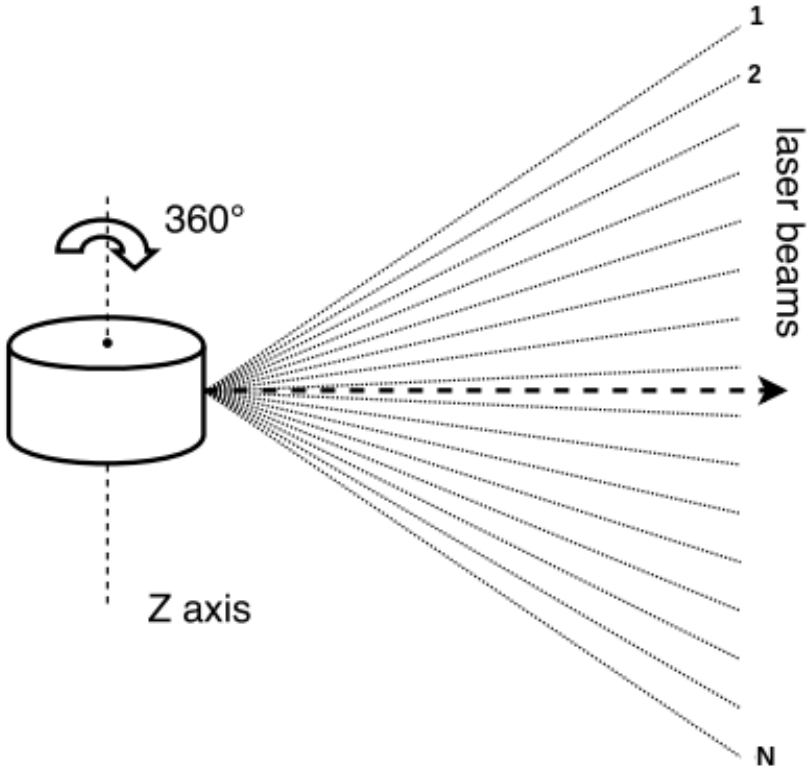

Figure 12: Velodyne LiDAR field of view. Adapted from Putkiranta (2020).

LiDAR beam, we define a 6D array of internal parameters, which is composed of internal 3D rotation and translation, then we add two extra parameters (additive and proportional correction) for the distance metric. 8 intrinsic parameters should be estimated for each beam, ($8 \times 16 = 128$ intrinsic parameters for Velodyne *VLP-16* LiDAR and $8 \times 128 = 1024$ for Velodyne *VLS-128*). Some of these parameters can be discarded, for example, in Equation (46), it can be seen that the roll angle is not used to compute the 3D point cloud. To this end, Muhammad et al. (2010) suggest incorporating the additive and proportional correction for distance metric, into extrinsic translation parameters, therefore, each LiDAR beam has 5 intrinsic parameters

$$\Theta_{intrinsic} = (\delta_{xi}, \delta_{yi}, \delta_{zi}, \alpha_i, \omega_i), \quad \forall i \in (0, \#Beams), \tag{49}$$

where $(\delta_{xi}, \delta_{yi}, \delta_{zi})$ are the internal displacement parameters and (α_i, ω_i) are the azimuth and elevation angles.

2.6.1 Calibration methods

Most of LiDAR sensors are provided with a factory calibration. However, the factory calibration is not always done perfectly, it may contain noise, or disturbance may

occur due to inaccurate transportation of the device. Lichti et al. (2000) argues that factory calibration parameters can often be improved.

There are several methods to perform LiDAR calibration, in this section we will briefly describe two of them.

1. Calibration based on known features - calibration targets are placed in some known positions in the environment. The optimization minimizes the error between the sensor estimation of the feature coordinates and their ground truth values. Typically, plane or cylinder features are used (Chan et al. 2013), due to the ease of detection and precise shapes.

2. **Data based calibration** - this approach does not require prior knowledge about the environment, but rather searches for local planarity or some keypoints descriptors in the point cloud. The features can be detected either automatically or manually, after which the calibration parameters are optimized to minimize the amount of noise in the observed features.

2.7 Camera-LiDAR extrinsics

Camera-LiDAR extrinsic calibration parameters are rotation **R** and translation **t** between sensor frames. In this section, we show the point correspondences registration

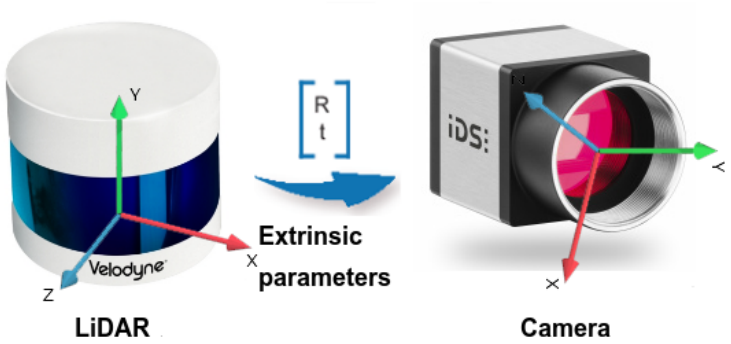

Figure 13: Camera-LiDAR extrinsic (rotation and translation) transformation.

and the transformation estimation. In the preprocessing phase, we extract the corners from the original point cloud and their 2D correspondences in camera space. A point cloud template is fitted in the resulted points to get the exact pose of each corner. Different fitting strategies may suffer from point cloud sparsity and the result can be biased towards higher density regions in the data. To overcome this issue, we propose to use only a carefully selected part of points (e.g., cloud margins) for fitting.

2.7.1 Convex hull

Convex hull is a method to acquire the shape around the data, it is a minimal set of points that forms a convex polygon around the given points. An example of a convex

hull is presented in Figure 14. Only hull vertexes (red points from the green line) are used for template fitting. The common method of computing the convex hull

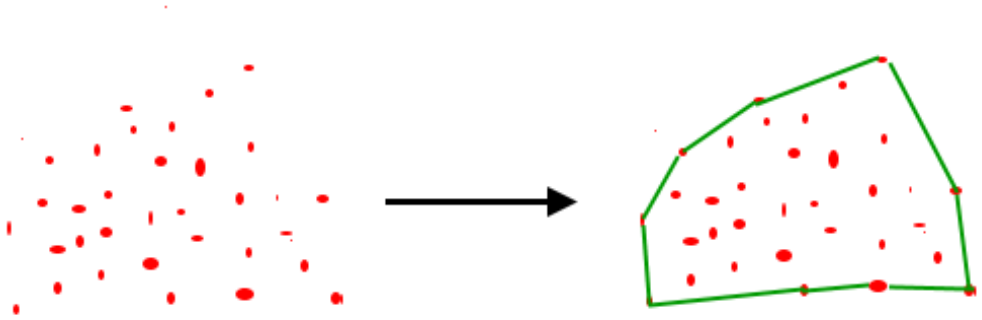

Figure 14: Example of convex hull.

around a given set of points is known as Graham's algorithm, presented by Graham et al. (1983). The main steps of the Graham's method are as follows:

1. Find the point with the lowest y coordinate, if there are 2 or more such points, take the one with the lowest x coordinate and push the point into a stack.
2. Sort the other $n-1$ points by the polar angle in counterclockwise order around the first point and push the second and third point into a stack.
3. For the remaining $n-3$ points, do the following:
   - Take the point at the top of the stack q, point next to top in stack p and current point r.
   - While the orientation of (p, q, r) is not counterclockwise, keep removing points from the stack.
   - Push the current point r into the stack.

The time complexity of the algorithm is $O(n \log n)$. The algorithm uses a stack data structure and the bottleneck is sorting the points by polar angle. A simple determinant is computed to find whether the points are oriented counterclockwise (make a left turn) or clockwise (right turn). The orient(...) function is used to sort the points. Figure 15 shows a visualization of orientation computed with det function.

$$\text{orient}(p, q, r) = \det \begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix} \tag{50}$$

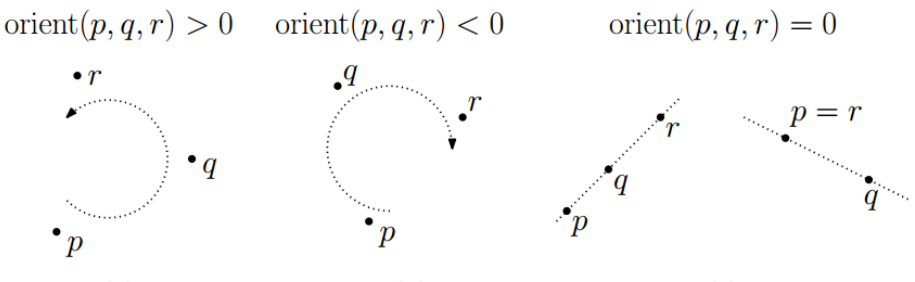

Figure 15: Orientation of the points computed with determinant function.

2.7.2 Iterative closest point

Iterative closest point (ICP) algorithm is a common method for point cloud fitting (Chetverikov et al. 2002). In the standard version of ICP, two point sets ($\mathbf{X} = \mathbf{x_1}, \ldots, \mathbf{x_n}$ and $\mathbf{P} = \mathbf{p_1}, \ldots, \mathbf{p_n}$) are given with known point correspondences. As shown in Figure 16, ICP estimates the transformation (rotation $\mathbf{R}$ and translation $\mathbf{t}$) from $\mathbf{X}$ to $\mathbf{P}$ that minimizes the sum of squared error

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{R}\mathbf{p}_i - \mathbf{t}||^2, \tag{51}$$

where $\mathbf{x}_i$ and $\mathbf{p}_i$ are point correspondences and N is the total number of points. If the point correspondences are known, we can compute the solution in a closed form.

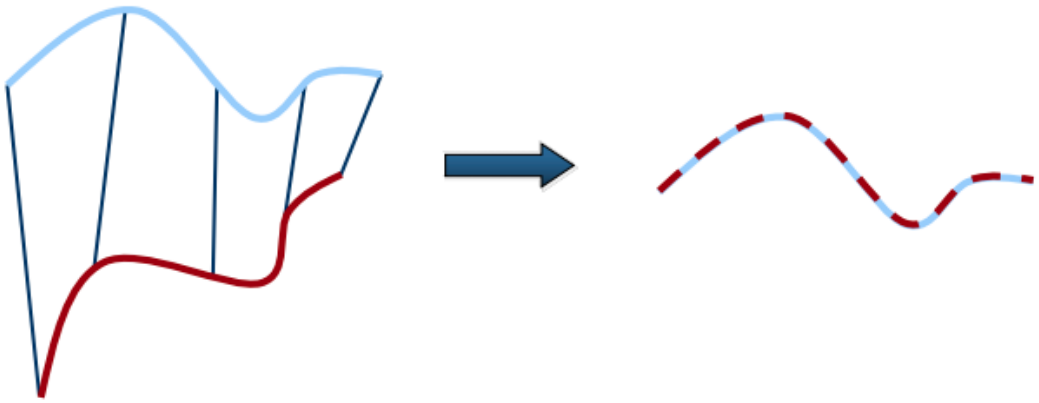

Figure 16: Iterative closest point (ICP) method, with known point correspondences. Adapted from Thrun et al. (2005).

We compute the mean of each data set

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i, \quad \mu_p = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i, \tag{52}$$

and center the data sets by

$$\mathbf{X}' = \mathbf{X} - \mu_x, \quad \mathbf{P}' = \mathbf{P} - \mu_p. \tag{53}$$

The covariance matrix is defined as

$$\mathbf{W} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}'_i\mathbf{p}'^{\top}_i, \tag{54}$$

with $\mathbf{x}'_i$ and $\mathbf{p}'_i$ points from the centered data sets. Taking SVD$(\mathbf{W}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$, leads to $\mathbf{R} = \mathbf{U}\mathbf{V}^{\top}$ and $\mathbf{t} = \mu_x - \mathbf{R}\mu_p$.

The key assumption of ICP method is that point clouds have the same size and point correspondences are known. As Besl et al. (1992) stated, if the point correspondences are unknown, it is generally impossible to find the optimal transformation between the data in one step. Data association with unknown point correspondences and data sparsity is the main problem of ICP. Different variants of ICP were proposed to handle the above specified problems:

1. **Point subsets** - the use of different sampling strategies (uniform, random, feature based) to select points from one or both point sets.

2. **Data association** - for each point in the first set, search for the closest point in the second set.

3. **Reject outlier points** - as described by Guo et al. 2011, select points from both point sets using RANSAC method.

2.7.3 Least squares pose estimation

The point correspondences are either 3D-2D (camera pixels and LiDAR) or 3D-3D (stereo camera and LiDAR). Usually, LiDAR is kept as a world frame and transformation is estimated relative to it. DLT algorithm, presented Section 2.4.1, can be used to estimate the transformation. However, DLT aims to estimate the whole projection matrix, which contains the internal camera calibration matrix $\mathbf{K}$ and extrinsic parameters, 11 DOF (5 for $\mathbf{K}$ matrix and 6 for camera pose). The intrinsic camera matrix $\mathbf{K}$ is already known, which can be leveraged to estimate the remaining unknown extrinsic parameters.

Pose estimation can be defined as a least square problem. Let $\mathbf{y} = \mathbf{f}(\mathbf{X})_{\mathbf{x}}$ where $\mathbf{f}$ is a function that returns the transformed point $\mathbf{X}$ and $\mathbf{x}$ is a vector of unknown

pose parameters with

$$\mathbf{x} = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ t_x \\ t_y \\ t_z \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} [x_1, y_1, z_1] \\ \vdots \\ [x_N, y_N, z_N] \end{bmatrix}, \tag{55}$$

where $(\theta_x, \theta_y, \theta_z)$ is the orientation and (t_x, t_y, t_z) correspond to position, N is the number of data points. The cost function can be defined as follows:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=0}^{\mathbf{N}} ||\mathbf{f}(\mathbf{X}_i)_{\mathbf{x}} - \hat{\mathbf{y}}_i||^2, \tag{56}$$

where $\hat{\mathbf{y}}$ is a vector of observed/ground truth points.

2.7.4 Perspective-n-Point

Perspective-n-Point (PnP) algorithm is another method to compute the camera pose (Lepetit et al. 2009). The standard version of the algorithm uses three points, known as P3P problem, presented by Gao et al. (2003). The result of this method can sometimes be inaccurate since it uses only three point correspondences. Moreover, the provided solutions are not unique, therefore, extra points are required. The camera projection matrix is defined as

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \quad \mathbf{RC}] = [\mathbf{KR}| - \mathbf{KRC}] = \mathbf{KR}[\mathbf{I}| - \mathbf{C}], \tag{57}$$

where $\mathbf{K}$ is the intrinsic camera calibration matrix, $\mathbf{R}$ corresponds to camera rotation, $\mathbf{C}$ is the center of the camera in world coordinate frame, and $\mathbf{I}$ is the identity matrix in 3D. A homogeneous 3D point can be projected into 2D space via

$$\lambda \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}. \tag{58}$$

Left hand side and right hand side of Equation (58) are equal, which means that their cross product is zero. Using the fact that the cross product of any vector and itself is zero and the cross product in a matrix form described in Equation (29), we can rewrite Equation (58) as

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \times \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = 0 \xrightarrow[\text{Eq (29)}]{\text{From}} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}_\times \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_\times \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = 0$$
$$\Rightarrow \underbrace{\begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix}}_{3\times3} \underbrace{\begin{bmatrix} \mathbf{X}^\top & 0_{1\times4} & 0_{1\times4} \\ 0_{1\times4} & \mathbf{X}^\top & 0_{1\times4} \\ 0_{1\times4} & 0_{1\times4} & \mathbf{X}^\top \end{bmatrix}}_{3\times12} \underbrace{\begin{bmatrix} \mathbf{P_1}^\top \\ \mathbf{P_2}^\top \\ \mathbf{P_3}^\top \end{bmatrix}}_{12\times1} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{3\times1}, \tag{59}$$

where $\begin{bmatrix} u & v & 1 \end{bmatrix}^\top$ is the image point, $0_{1\times4}$ is a 1×4 block of zeros and $\mathbf{P}_i$ is the row i from projection matrix $\mathbf{P}$. There are 12 unknown parameters and every point correspondences introduces two constraints. The minimum of six points are required to solve the equation. For given six points, Equation (59) can be expressed as 18×12 matrix $\mathbf{A}$

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & -X_1 & -Y_1 & -Z_1 & -1 & X_1v_1 & Y_1v_1 & Z_1v_1 & v_1 \\ X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1u_1 & -Y_1u_1 & -Z_1u_1 & -u_1 \\ -X_1v_1 & -Y_1v_1 & -Z_1v_1 & -v_1 & X_1u_1 & Y_1u_1 & Z_1u_1 & 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -X_n & -Y_n & -Z_n & -1 & X_nv_n & Y_nv_n & Z_nv_n & v_n \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -X_nu_n & -Y_nu_n & -Z_nu_n & -u_n \\ -X_nv_n & -Y_nv_n & -Z_nv_n & -v_n & X_nu_n & Y_nu_n & Z_nu_n & 1 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}:18\times12} \cdot \underbrace{\begin{bmatrix} \mathbf{P_1}^\top \\ \mathbf{P_2}^\top \\ \mathbf{P_3}^\top \end{bmatrix}}_{12\times1} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{18\times1}.$$

Using SVD($\mathbf{A}$) $= \mathbf{U\Sigma V}^\top$, the last column of $\mathbf{V}^\top$ gives the solution for the unknowns $\begin{bmatrix} \mathbf{P}_1^\top & \mathbf{P}_2^\top & \mathbf{P}_3^\top \end{bmatrix}^\top$. The $\mathbf{P}$ matrix is further decomposed into $\mathbf{R}$ and $\mathbf{t}$

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \xrightarrow[\mathbf{K^{-1}}]{Multiply \quad with} \underbrace{\mathbf{K^{-1}}}_{3\times3}\underbrace{\mathbf{P}}_{3\times4} = \underbrace{[\mathbf{R}|\mathbf{t}]}_{3\times4} \Rightarrow \mathbf{R} = \mathbf{K^{-1}P}_{1:3}, \tag{60}$$

where $\mathbf{P}_{1:3}$ are the first three columns of the projection matrix $\mathbf{P}$ and $\mathbf{R}$ is the rotation matrix, which should be orthonormal. To enforce orthonormality, we again take the SVD($\mathbf{R}$) $= \mathbf{U\Sigma V}^\top$ and the new rotation matrix is $\mathbf{R}_+ = \mathbf{UV}^\top$. Similarly for translation, $\mathbf{t} = \frac{\mathbf{K^{-1}P}_4}{\delta_1}$, where $\mathbf{P}_4$ is the last column of the projection matrix and δ_1 is the first eigenvalue of the SVD($\mathbf{R}$). This is one method to solve PnP problem, but also another techniques do exist. Lepetit et al. (2009) presents efficient version of PnP which returns an unique solution with at least four point correspondences. Usually, more point correspondences are used to reduce the impact of noisy data.

3 Research material and methods

In this section, we present the hardware and software system that was used in this work, as well as data collection and processing techniques.

3.1 Research platform

All experiments were performed on a research platform built on top of the Ford Mondeo vehicle presented in Figure 17. The research platform is called Autonomous Research Vehicle Observatory (ARVO) and it is equipped with five LiDAR sensors. The main LiDAR Velodyne *VLS-128* is located in the center of the roof and four Velodyne *VLP-16* LiDARs are installed in each corner of the vehicle's roof. Details on LiDAR sensors are provided in LiDAR Velodyne (2018) datasheet. Two monochrome

Figure 17: Autonomous-capable Ford Mondeo Hybrid research platform.

and three RGB cameras are installed inside the vehicle, the installation is presented in Figure 18. A thermal camera is installed on the left side of the vehicle's roof. The sensors measurements are processed by an Intel i7 2.9 GHz processor from a mobile workstation with 64 GB of memory embedded in the vehicle. All software systems run under the Ubuntu 16.04 and Robot Operating System (Quigley et al. 2009).

In this thesis, two monochrome cameras (IDS GmbH 2021) and the Velodyne VLS-128 LiDAR were used. The selected monochrome camera sensors provide images with 1936×1216 pixel resolution and up to 166 FPS. As shown in Figure 19, a large baseline between cameras reduces the depth uncertainty, therefore, the distance between the monochrome cameras is set to 96 cm. All experiments were performed

with monochrome cameras, but the provided solution can also be used with RGB and thermal cameras.

Figure 18: RGB and Monochrome cameras installed inside the vehicle.

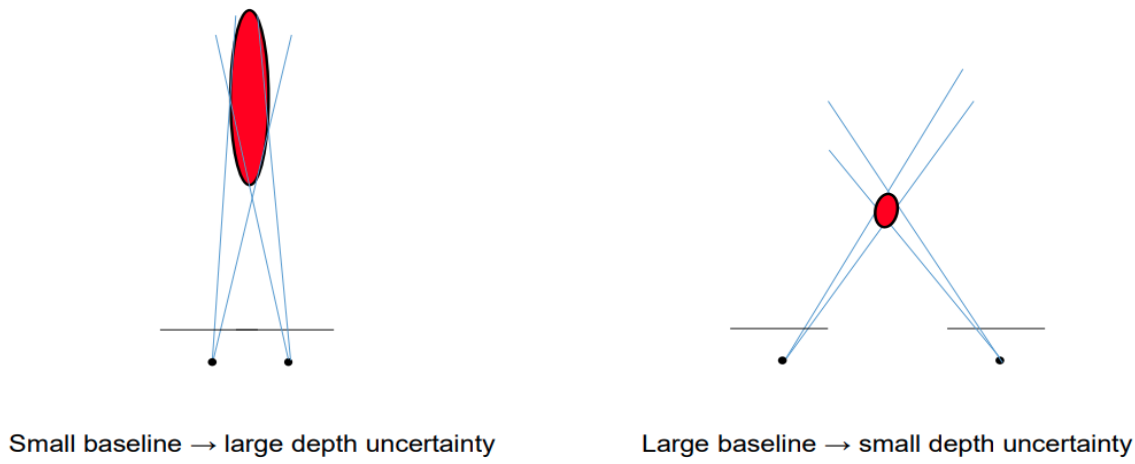

Figure 19: Effect of small vs. large baseline. A larger baseline minimizes the depth uncertainties (red ball). Adapted from Yousif et al. (2015).

3.2 Mono and stereo camera data collection

Point correspondences between 3D environment and their 2D pixel location are collected with two calibration targets, a ChArUco (Gwon et al. 2018) and a chessboard presented in Figure 21. Using a chessboard, it is complicated to collect data points from the very edges of the images and it is preferable to have those data points, because they constrain the camera lens distortion properly. In OpenCV module, the entire chessboard must be visible in the camera FOV, ChArUco board was used to

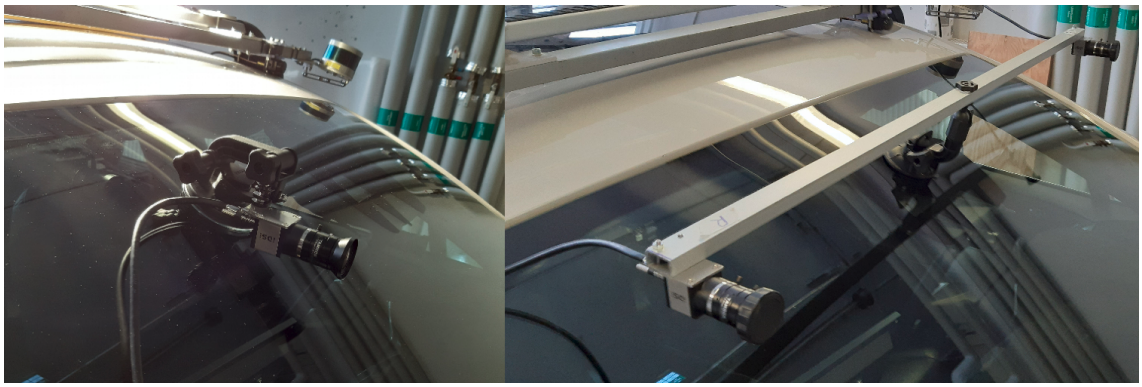

Figure 20: Camera sensors for mono (left) and stereo (right) case installed outside the vehicle, used to collect data that is not affected by the windshield.

overcome this disadvantage. The benefit of using ChArUco markers is that each cell is uniquely identified and can provide enough information to obtain the camera pose. As can be seen in the top side of Figure 22, the ChArUco board allows us to collect data points from the very edges of the image. We used two monochrome cameras of the same type, however, we treated them individually, therefore, two separate data sets were collected. For the calibration targets used in this thesis, the square size of the chessboard is 10 cm, while the marker of the ChArUco is 6 cm, therefore, the chessboard can be detected further away from the camera. To increase the diversity of the recorded data set, data points collected with chessboard and ChArUco targets are combined. The more point correspondences from different configurations, the better. To estimate the windshield effect on camera calibration, we collected images while the cameras were installed inside and outside the car, the setup is presented in Figures 18 and 20. Similarly, we collected the data sets for stereo calibration.

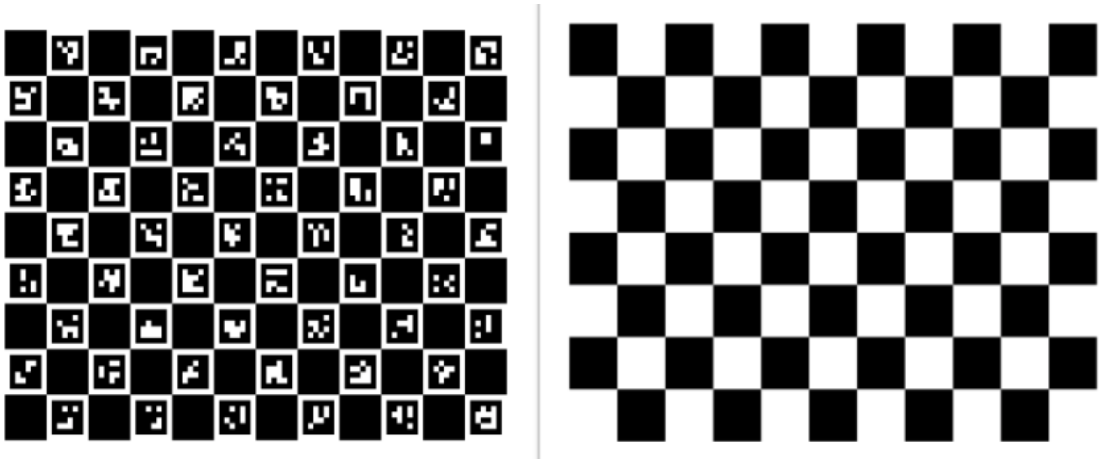

Figure 21: On the left ChArUco 8×11 board with marker size 6 cm. On the right 7×10 chessboard with square size of 10 cm.

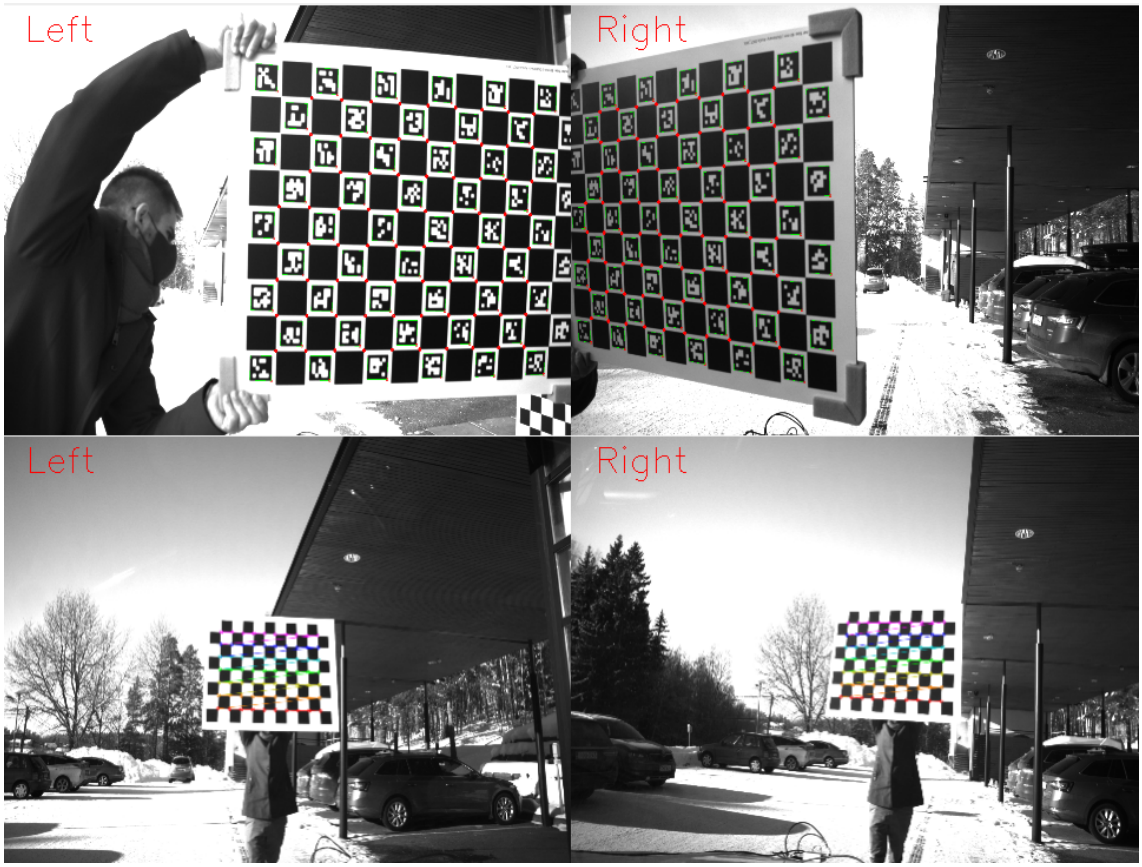

Figure 22: Top: point correspondences with ChArUco calibration board for both cameras, Bottom: point correspondences with standard chessboard.

In the initial setup, the camera is static and the calibration board is placed in different locations and orientations, to accumulate as many variations as possible. For simplicity, we can assume that the calibration board is static in XY plane and that the camera moves around it, bringing Z-axis always to zero. In this case for X and Y coordinates of the 3D points we defined values: $(0, 0), \ldots, (i, j), \ldots, (N_{cols-1}, N_{rows-1})$, where (N_{cols}, N_{rows}) are the number of corners per columns and rows of the calibration board. The data points are in scale of the calibration board square size (10 cm for the chessboard and 6 cm for the ChArUco board). 3D points are multiplied with square size, to convert them to a scale of 1 cm. OpenCV was used to detect and extract the 2D corners from the image. Table 1 shows details on camera calibration data set. Each image recorded with the chessboard pattern provides 70 points (7×10 corners), while the number of points recorded with ChArUco is variable, since the whole board might not be visible in the image. The number of images collected with the ChArUco board is higher than the number of images collected with the chessboard, so that in the end we could obtain approximately equal number of point correspondences.

Table 1: The number of images recorded for mono and stereo calibration, for each camera inside and outside the car. Data collection was performed with both calibration boards.

	Mono dataset			
	Chessboard		**ChArUco**	
Cams	Inside	Outside	Inside	Outside
Left camera	48	61	106	99
Right camera	39	62	113	101
	Stereo dataset			
	Chessboard		**ChArUco**	
Inside	30		70	
Outside	45		71	

3.3 Camera-LiDAR synchronisation

Camera and LiDAR have their own internal clock. Cameras are set to send messages at a frequency of 20Hz, while LiDAR is set to publish massages at a rate of 10Hz.

Synchronisation is required to associate measurements from different sensors. Software-based sensor synchronization package provided by ROS filters messages by a given time threshold. But the time threshold may vary, so, hardware-based synchronisation is a better solution. In this thesis, an external clock was used to synchronise sensors messages. A Pulse Per Second (PPS) signal is sent by GNSS receiver and its value is recorded by the cameras together with the triggered images. To associate camera and LiDAR messages, all the messages in a two second interval are collected in a buffer. Every LiDAR point has the time information when it was recorded. Therefore, we extracted the time series by taking a modulo operation of each LiDAR point time and time interval between 2 PPS signals. For camera time series, we divided the 2 seconds interval between the number of messages. For each LiDAR point, only the closest camera message was used. This method downscales the camera's rate to match the LiDARs rate. Figure 23 shows the time series alignment between the LiDAR and camera messages.

3.4 Camera-LiDAR data collection

As stated by Luo et al. (2002), there are three levels of information fusion that can be grouped as low-level (i.e., raw data), mid-level (i.e., features), and high-level (i.e., object-based fusion). In this thesis, mid-level data is used to estimate the extrinsic transformation between sensor frames. This subsection presents the techniques that were applied to collect and extract the feature point correspondences from camera and LiDAR data.

Besides the standard chessboard, we used a custom calibration target presented on the right side of the Figure 24. This target has four printed ArUco (Ferrão

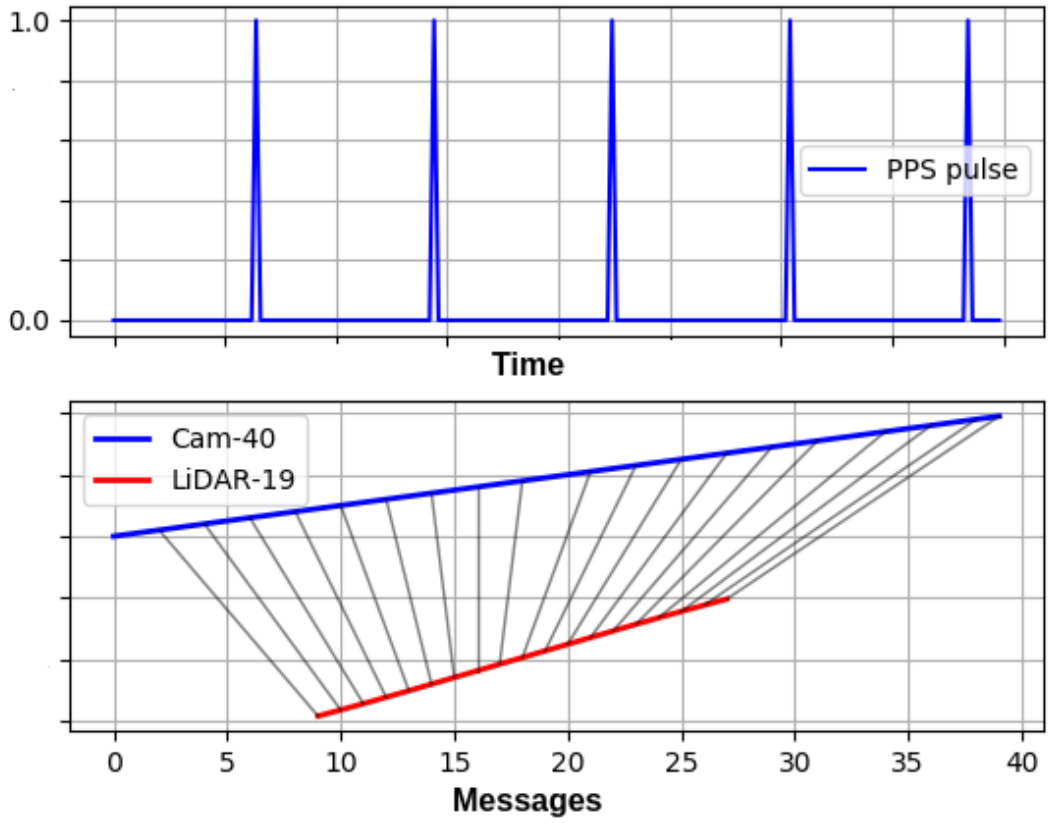

Figure 23: Camera-LiDAR synchronisation. Top: GNSS pulse, Bottom: LiDAR and Camera time series. PPS pulse is 1 every 2 seconds, in this interval ≈ 40 camera messages and ≈ 19 LiDAR messages are collected and later synchronized. The synchronization is initiated when the PPS pulse is 1, after which the buffer is cleared.

et al. 2018) markers, to be easily detected on the camera. In the LiDAR frame, it can be detected as a plane and a circle inside it. The dimension of the board is 1×1 meter and the circle diameter is 33 cm. Each pair of messages provide camera image points and LiDAR point cloud correspondences. Figure 24 shows an example of the camera and LiDAR point cloud data of the same view. The point correspondences in camera space are chessboard corners and plane corners with circle center, for the second board. The same 3D point correspondences are detected in the original point cloud. RANSAC method was used to extract the plane points from the original point cloud.

3.4.1 RANSAC plane fitting

RANSAC stands for random sampling and consensus (Yaniv 2010). It is an iterative method used to fit a model into a data that contains outliers. We used RANSAC to fit a plane in a 3D point cloud.

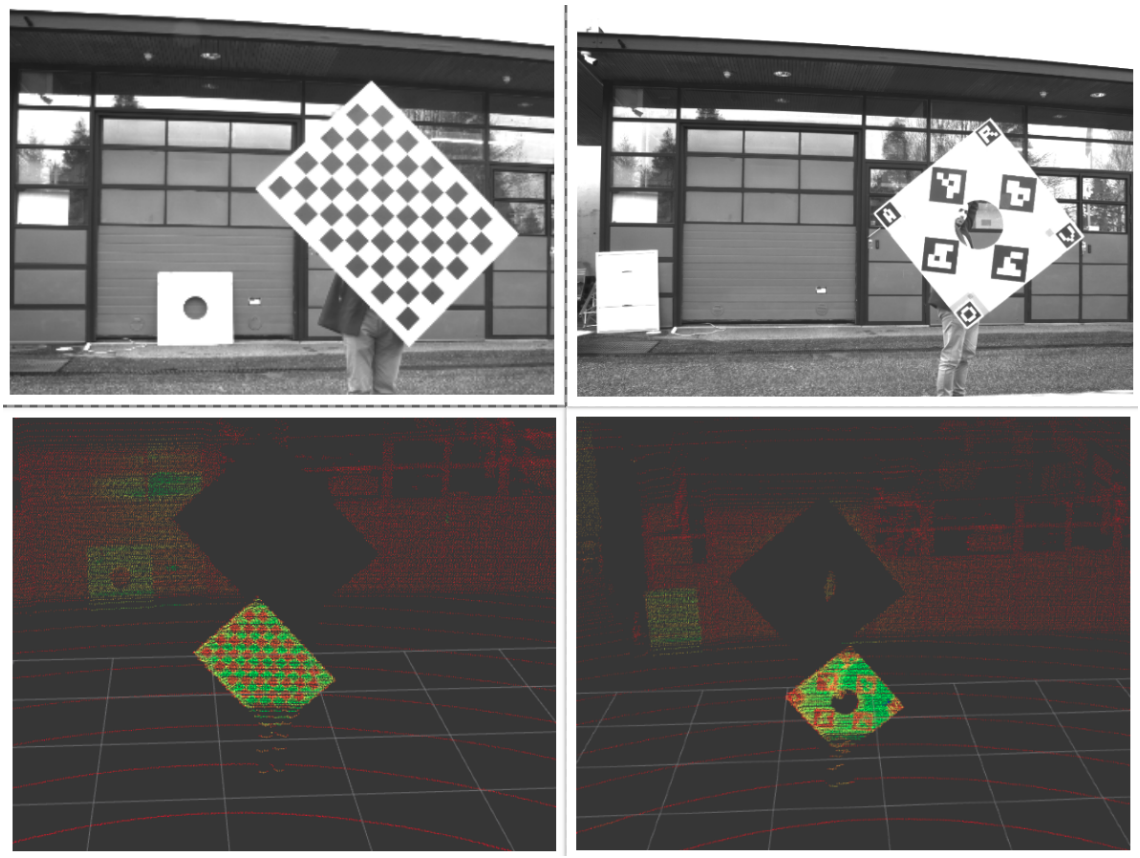

Figure 24: Camera-LiDAR view with 2 calibration boards. The colour of the point cloud is based on point intensity.

According to Gallo et al. (2011) the main steps of the algorithm are:

1. Randomly select a minimal subset of points to fit a model. For a 3D plane, we need at least 3 points. The following equation shows the plane model in cartesian form:
$$ax + by + cz + d = 0, \tag{61}$$
where (x, y, z) is a 3D point and (a, b, c, d) are plane parameters.

2. Fit the model into randomly selected points. Given plane equation and 3 points we can estimate the parameters (a, b, c, d) using:
$$\begin{aligned} a &= [(y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_2)], \\ b &= [(z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_2)], \\ c &= [(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_2)], \\ d &= -(ax + by + cz). \end{aligned} \tag{62}$$

3. For all $n-3$ points, compute their deviation from the estimated plane using:

$$\text{dist}_i = \frac{ax_i + by_i + cz_i + d}{\sqrt{a^2 + b^2 + c^2}} \tag{63}$$

   where n is the total number of points and dist_i is the deviation for point i

4. If the estimated dist_i is within a given threshold, save the point as an inlier, otherwise save it as outlier.

5. Save the (a, b, c, d) parameters with the max number of inliers.

6. Repeat the steps 1-5 M times, where M is max number of iterations.

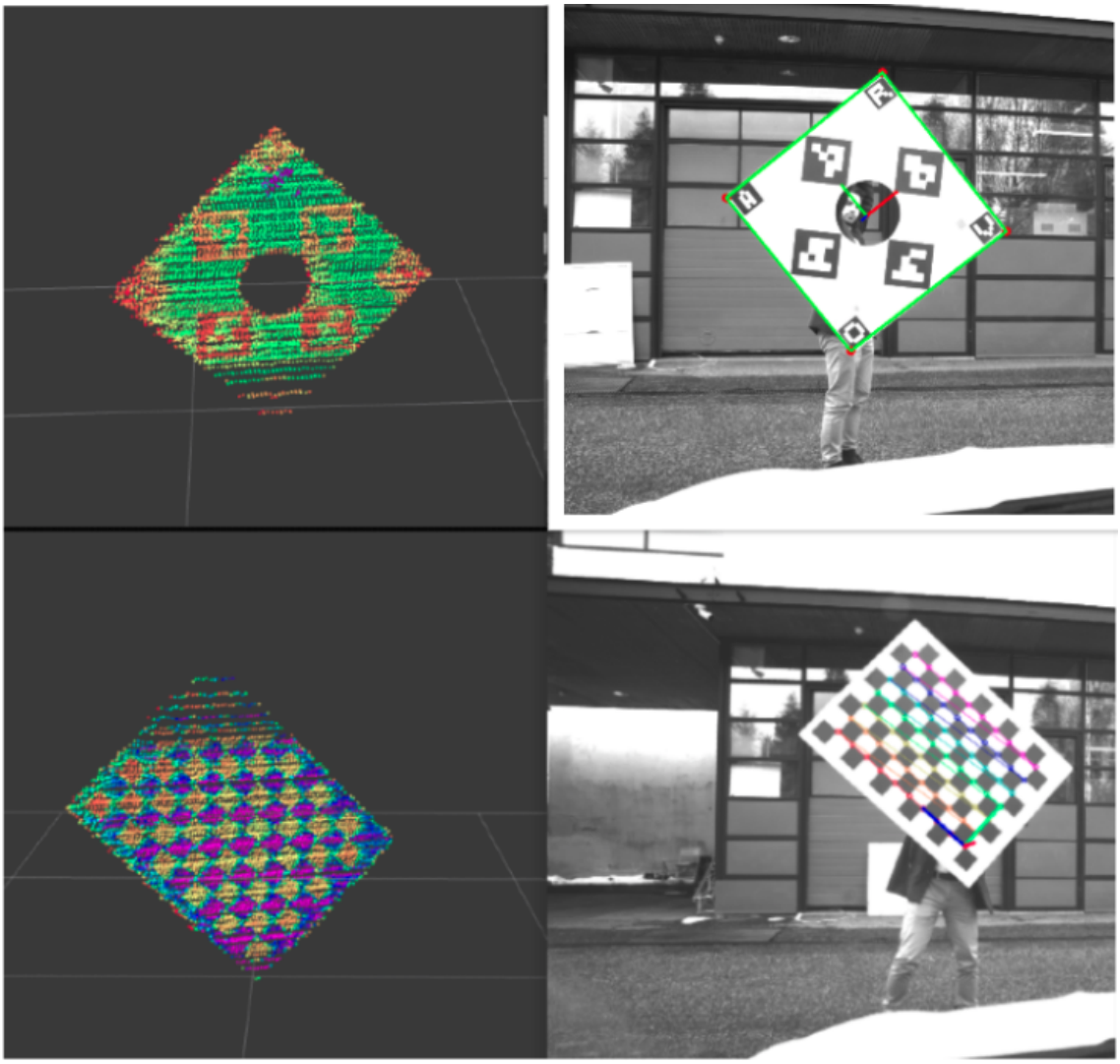

Figure 25: Camera-LiDAR filtered point cloud with RANSAC and camera calibration targets.

The result of RANSAC plane fitting is presented in Figure 25. All points that do not lie on the calibration target are treated as outliers. As we can see from Figures 24 and 25, the point cloud has sparse regions, due to LiDAR beam structure. We fit a board model into resulted plane to extract the corners. Data sparsity causes problems in model fitting because the solution is biased towards higher data density. To overcome this issue, margins of the boards were extracted and fitted with the convex hull algorithm, presented in Section 2.7.1. The board's margins are composed of the first and the last point from each LiDAR ring that lie on the target. Therefore, to extract the borders, the points were grouped by their ring information. Similarly, the points that correspond to the inside circle margin, were extracted. Given a distance threshold, we checked if the distance between two consecutive points from the same ring is bigger than the threshold, then the point belongs to the inside circle. Given a set of points that belong to a circle, we use least squares method to estimate the circle parameters.

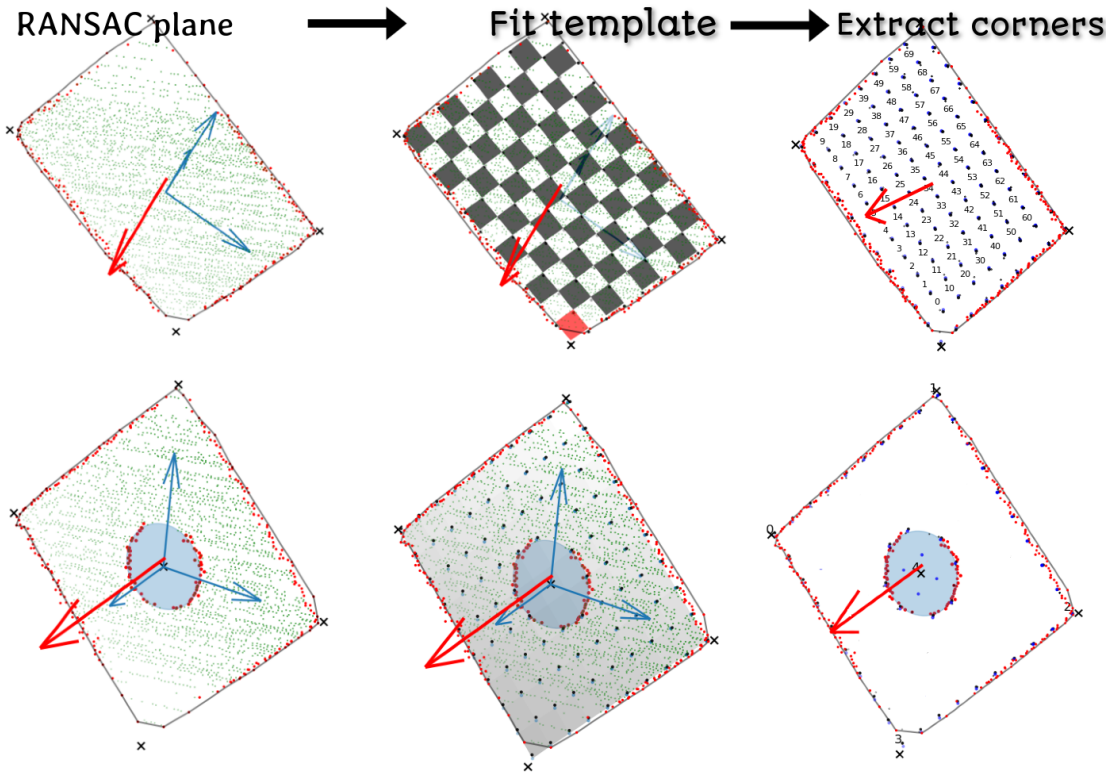

Figure 26: Point correspondences extraction from LiDAR point cloud. Top-chessboard plane, Bottom-custom calibration plane. 1) Filter point cloud with RANSAC and apply convex hull on margin points, 2) Fit a template into the filtered plane, 3) Extract corners from the point cloud.

3.4.2 Least squares circle fitting

As shown in Figure 26, compared to the chessboard, the custom designed board provides only four points (the corners). To improve the model fitting accuracy, we decided to fit a circle in the center of the board which will result in an extra feature. The equation of circle is

$$(x - x_0)^2 + (y - y_0)^2 = r^2, \tag{64}$$

where (x_0, y_0) is the center of the circle, r is the radius and (x, y) is a point that belongs to the circle. Using $(a - b)^2 = a^2 - 2ab + b^2$, Equation (64) becomes:

$$\begin{aligned} x^2 - 2xx_0 + x_0^2 + y^2 - 2yy_0 + y_0^2 = r^2 \rightarrow \\ 2xx_0 + 2yy_0 + r^2 - x_0^2 - y_0^2 = x^2 + y^2. \end{aligned} \tag{65}$$

The above equation can be re-written in a matrix form:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 2x_0 \\ 2y_0 \\ r^2 - x_0^2 - y_0^2 \end{bmatrix} = \begin{bmatrix} x^2 + y^2 \end{bmatrix} \rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}}_{unknowns} = \begin{bmatrix} x^2 + y^2 \end{bmatrix}. \tag{66}$$

For multiple observations Equation (66) becomes:

$$\begin{bmatrix} x_i & y_i & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} x_i^2 + y_i^2 \end{bmatrix}. \tag{67}$$

If we have N data points, we can stack the Equation (67):

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ & \vdots & \\ x_N & y_N & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}}_{\mathbf{p}} = \underbrace{\begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \vdots \\ x_N^2 + y_N^2 \end{bmatrix}}_{\mathbf{b}}. \tag{68}$$

The system becomes:

$$\mathbf{Ap} = \mathbf{b}, \tag{69}$$

which can be solved by minimizing $||\mathbf{Ap} - \mathbf{b}||^2$, therefore, using pseudo inverse

$$\mathbf{p} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}. \tag{70}$$

ICP method presented in Section 2.7.2, was used to fit the cloud template into plane points. Figure 26 shows the steps and results of point correspondences extraction from the point cloud plane. Two types of point correspondences were extracted: 2D-3D that correspond to the mono camera and LiDAR frames, and 3D-3D that correspond to stereo cameras (3D reconstructed boards) and LiDAR. We computed stereo 3D points using depth from disparity technique, presented in Section 2.5.3. Table 2 provide the information about the collected data set for camera and LiDAR point correspondences. Every camera-LiDAR view recorded with chessboard provides 70 point correspondences, while the custom target provides only 5 points (4 corners and circle center).

Table 2: The number of images and point clouds correspondences recorded for camera-LiDAR calibration. The data set was recorded with two calibration targets (chessboard and custom board). Each chessboard view provids 70 points. Each custom board view provides 5 points.

	Camera-LiDAR dataset	
	Chessboard	**Custom-board**
Images	18	23
Data points	1260	115

3.5 Data analysis and processing

3.5.1 Mono and stereo camera

The pixel size of the mono IDS GmbH 2021 is 5.86 μm and the pixels are square, this means that the focal length in pixels in x and y direction should be the same. As described in Section 2.4.2, we calibrate mono cameras using modified DLT algorithm with 3D-2D point correspondences. Each camera is separately calibrated

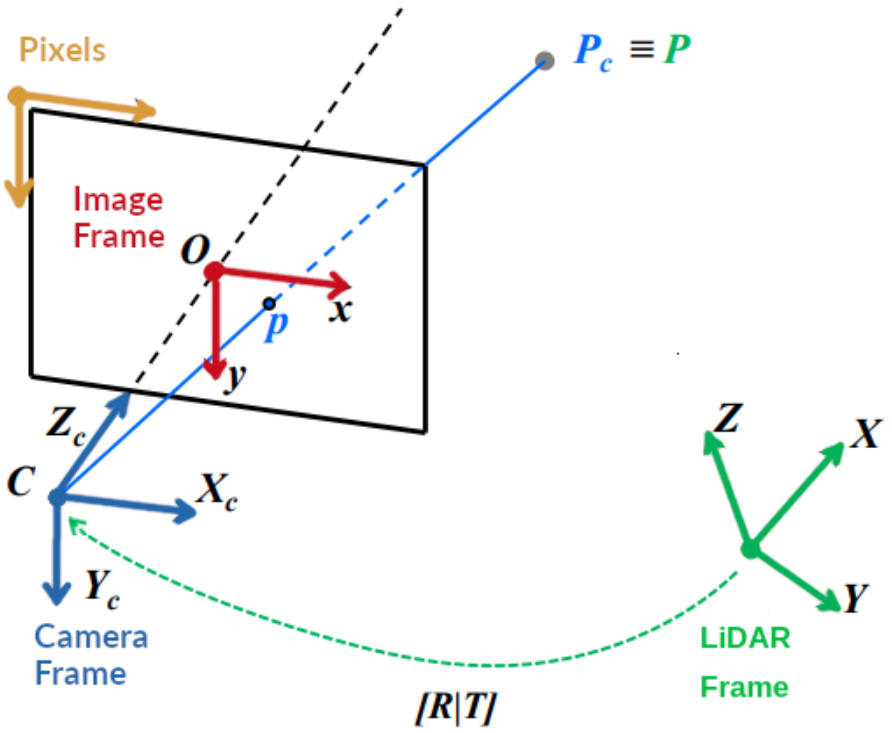

Figure 27: Camera-LiDAR transformation. LiDAR points are converted to camera frame using camera-LiDAR extrinsic parameters, and then points are projected on the image using camera intrinsic parameters. Adapted from Scaramuzza et al. (2011).

with chessboard and ChArUco data sets to check if different calibration target brings any improvements. Similarly for the windshield effect, cameras are calibrated inside and outside the car, with the chessboard, ChArUco and combined (chessboard + ChArUco) point correspondences. A similar process was performed for stereo camera calibration. *StereoCalibrate* function available in OpenCV, was used to compute rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the cameras. Calibration was realized with all distortion models: rational (RAT), thin prism (THP), tilted (TIL), and complete (CMP), described in Section 2.3.

3.5.2 Camera-LiDAR transformation

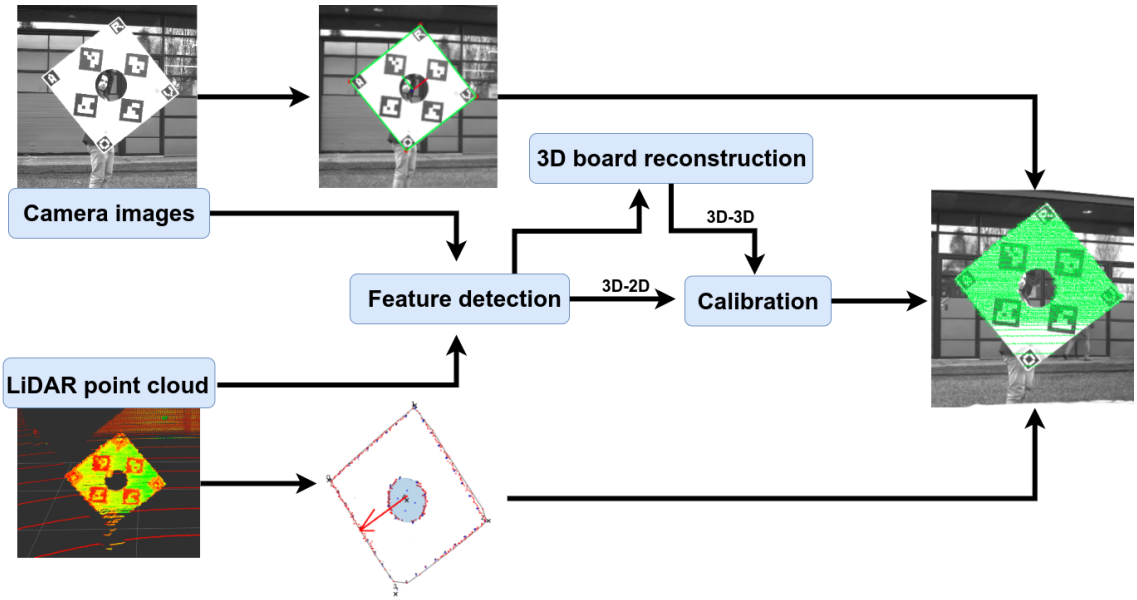

Figure 28: Camera-LiDAR extrinsic calibration steps. Feature detection is applied to the original data set. Calibration can be done either with 2D-3D or 3D-3D point correspondences.

Camera-LiDAR extrinsic parameters are used to rotate and translate the points from LiDAR frame to camera frame. The calibration process is presented in Figure 28. PnP algorithm, described in Section 2.7.4, was used to find the transformation between the 2D-3D point correspondences. We kept the LiDAR frame as the origin and estimated the transformation relative to it. In the 3D-3D approach, we estimated the transformation between the LiDAR and stereo cameras using ICP and least squares technique presented in Section 2.7.3. One disadvantage of this technique is that calibration based on 3D-3D point correspondences takes an extra step to reconstruct the 3D camera-based point cloud before calibration.

We computed the transformation between the LiDAR and each camera individually, therefore, we can validate stereo camera extrinsic calibration parameters. As

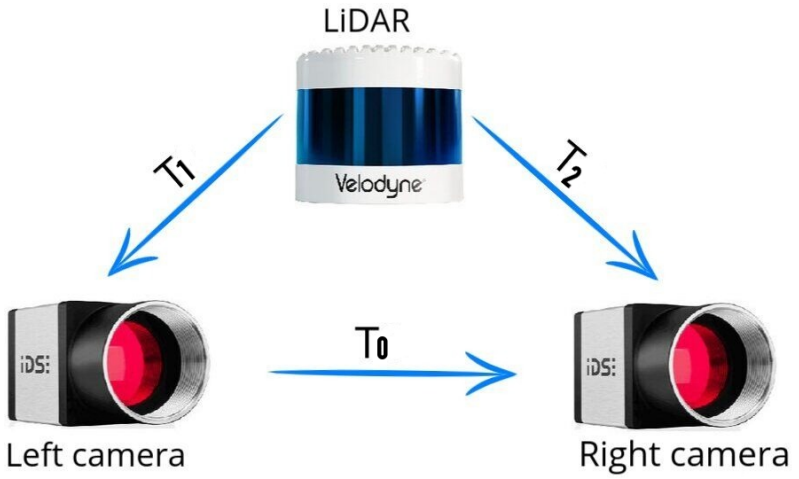

Figure 29: Stereo camera and LiDAR frames transformations.

shown in Figure 29, $\mathbf{T}_0$ can be inferred in terms of $\mathbf{T}_1$ and $\mathbf{T}_2$.

$$\mathbf{T}_0 = \mathbf{T}_1^{-1}\mathbf{T}_2 \tag{71}$$

3.5.3 Occlusion handling

Camera-LiDAR occlusion issue appears because of different sensor locations. The same scene is viewed from a different angle. As shown in Figure 30, some blue points are occluded in camera view, and for this reason, the blue 3D point will be coloured in red. To overcome this issue Schneider et al. (2010) propose to perform 3D point cloud clustering, sort the cluster based on their depth and group the projected points by their cluster. This method treats the points as groups and enforces the points from the same group to have the same depth. We propose a different approach to

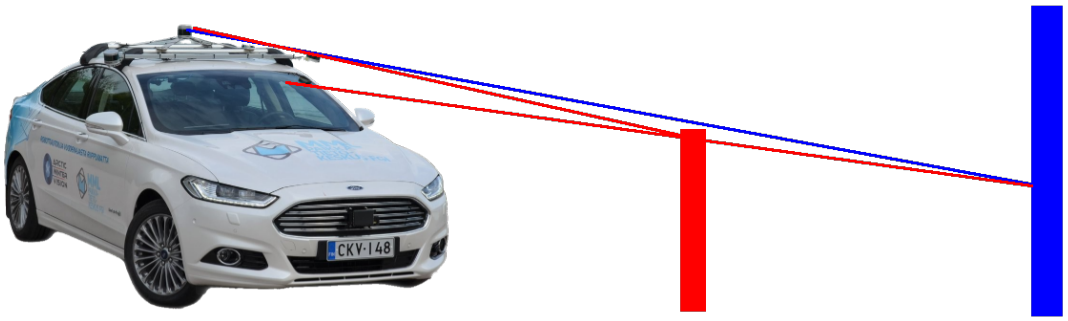

Figure 30: LiDAR and camera have different locations, therefore, both sensors have different view. Some of the blue points are visible by LiDAR, but not visible by camera, these points are occluded. The colour of the occluded blue point will be red since the camera cannot see the actual colour of the point.

handle the occlusion problem. Based on Figure 30, some occluded blue points may be projected in the same pixel locations as red points or nearby red pixels. Therefore, to remove occlusion, we filter the data in image space. The following algorithm assumes the extrinsic calibration between the camera and LiDAR to be done. Our idea is to treat each point as a rectangle and set the points that lie inside that rectangle as occluded if they have different depth. The pseudocode is presented in Algorithm 1. The outcome of the method are further discussed in the results Chapter 4.

Algorithm 1 Occlusion removal

```
Define box size b_x, b_y, depth threshold t and neighbours k
Project the point cloud in the image space
for point ∈ point cloud do
    compute k nearest neighbours for point in 2D pixels space
end for
Sort the projected point cloud by depth
for point ∈ projected point cloud do
    Create a box centered in point and assign it to current point
end for
for point ∈ projected point cloud do
    Check if point falls inside any rectangle of its neighbours
    Compute depth difference between point and its neighbours
    if point lies inside neighbour rectangle and depth difference is greater than t
then
        Set point as occluded
    end if
end for
Return the points that aren't occluded
```

4 Results

This chapter presents the results and answers to the research questions.

4.1 Mono and stereo camera calibration

Mono and stereo camera calibration results are compared based on the root mean squared error (RMS), the number of data points, and the aspect ratio of the focal length in pixels in x and y directions. A small RMS error is a required outcome, but not enough. For example, calibration based on fewer data points may result in a smaller re-projection error, which means that camera model will be good for those few data points. However, if we have a large enough data set, a small RMS error is interpreted as a good result. The RMS error is computed as follows

$$\text{RMS} = \sqrt{\frac{\sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2}{N}}, \tag{72}$$

where N is the number of data points, $\mathbf{x}_i$ is the observed pixel point and $\hat{\mathbf{x}}_i$ is projected pixel, using projection matrix from Equation (8).

4.1.1 Calibration with different calibration boards

We calibrated the cameras using two calibration boards (ChArUco and chessboard). Table 3 presents left camera calibration outside the car with both calibration targets. The results with both boards are similar, with small differences of 2-3 pixels for intrinsic parameters and 0.06 pixels for re-projection error. We concluded that there is no significant performance of one target over another. However, the advantage of the ChArUco board is the easiness to collect data points from the very edges of the frames, since it does not require to be fully visible in the camera FOV. The final intrinsic parameters are estimated based on the combined points collected with chessboard and ChArUco.

Table 3: Left camera calibration outside the car with ChArUco-99 images (7543 points) and Chessboard-61 images (4270 points). Values are in pixels.

	params	**ChArUco**	**Chessboard**
1	f_x	1368.3	1364.3
2	f_y	1368.1	1365.0
3	p_x	966.1	968.2
4	p_y	603.7	605.0
5	RMS	0.21	0.15

Table 4: Outside right camera calibration based on 11927 point correspondences, with all available distortion models. Values are in pixels.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	f_x	1367.5	1367.2	1367.5	1367.5	1367.3	1367.5	1367.3	1367.3
2	f_y	1367.4	1367.2	1367.5	1367.5	1367.3	1367.5	1367.3	1367.3
3	p_x	953.0	953.0	954.3	954.3	954.3	960.8	947.9	960.8
4	p_y	610.5	610.5	616.1	616.1	616.18	615.7	613.3	615.7
5	s_k	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	RMS	0.172	0.171	0.171	0.171	0.171	0.171	0.171	0.171

Table 5: Outside left camera calibration based on 11813 point correspondences, with all available distortion models Values are in pixels.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	f_x	1367.3	1367.0	1367.4	1367.4	1367.1	1367.3	1367.0	1367.0
2	f_y	1367.4	1367.1	1367.5	1367.5	1367.2	1367.5	1367.2	1367.2
3	p_x	966.2	966.2	975.5	975.5	976.7	965.5	967.5	965.6
4	p_y	604.2	604.3	610.1	610.1	611.7	607.3	607.2	607.3
5	s_k	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	RMS	0.200	0.199	0.200	0.200	0.199	0.200	0.199	0.199

4.1.2 Windshield effect on camera calibration

As mentioned earlier, we refer to *inside* and *outside* for the calibration done while the cameras are installed inside or outside the car. To estimate the windshield effect, we calibrated both cameras inside and outside the car. The outside calibration is not affected by the windshield, for this reason, these results will be used as a basis for comparison. The left camera is used as a world frame, and the right camera is transformed relative to it. Tables 4 and 5 show the outside calibration for both cameras. Calibration was performed with all distortion models. For all cases, we got the same focal length, $f_x \approx f_y \approx 1367.5$ pixels. The use of several distortion models for outside calibration does not bring any significant improvement. Inside calibration for left and right camera are presented in Tables 6 and 7. We noticed that focal lengths f_x and f_y are no longer the same. The windshield introduces noise that increases the focal length mostly in y-direction and shifts the principal point, down on x and y axis. Also, the re-projection error is higher. On the contrary with outside calibration, we noticed that different distortion models can handle the windshield noise. For example, the best *inside* calibration is achieved with complete (CMP) distortion model, which gives the smallest re-projection error (0.31 for the right camera and 0.33 for the left camera) and focal length parameters that are closer to values estimated *outside* the car. For a full mono camera calibration parameter set, see Appendix 7. Table 8 shows stereo calibration results. We got similar results

Table 6: Inside right camera calibration based on 11668 point correspondences, with all available distortion models. Values are in pixels.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	f_x	1367.57	1367.7	1374.12	1374.12	1373.9	1368.8	1372.1	1369.1
2	f_y	1373.76	1373.9	1382.3	1382.3	1382.2	1375.8	1387.13	1376.0
3	p_x	955.7	955.6	863.2	863.2	863.6	990.1	711.5	988.8
4	p_y	601.9	601.9	502.0	502.0	501.5	691.8	653.4	691.3
5	s_k	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	RMS	0.520	0.519	0.347	0.347	0.347	0.318	0.378	0.317

Table 7: Inside left camera calibration based on 11596 point correspondences, with all available distortion models. Values are in pixels.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	f_x	1372.0	1371.52	1373.73	1373.73	1373.1	1366.8	1369.4	1366.5
2	f_y	1377.9	1377.5	1379.7	1379.7	1379.3	1370.1	1375.5	1369.7
3	p_x	966.3	966.1	1059.3	1059.3	1057.0	970.8	1033.1	965.5
4	p_y	589.1	589.0	524.3	524.3	522.8	599.6	632.6	602.0
5	s_k	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	RMS	0.430	0.427	0.364	0.364	0.362	0.339	0.408	0.337

as in the mono calibration case, bigger RMS error inside the vehicle. However, the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the left and right camera, are almost the same inside and outside. We concluded that the windshield affects both cameras in the same way, therefore, the transformation between the cameras remains the same.

4.2 Camera-LiDAR extrinsic calibration

The extrinsic calibration based on 3D LiDAR and 2D pixel point correspondences was performed with PnP method, described in Section 2.7.4. Table 9 shows the extrinsic parameter between the LiDAR and the left camera, and Table 10 shows the transformation between the LiDAR and the right camera. Results obtained with both targets are approximately the same. For the final estimation, points collected with chessboard and with custom-board were merged. Figure 31 shows the projection from LiDAR to image frame. The extrinsic parameters estimated with 3D-3D point correspondences, are presented in Table 11. Camera-based 3D points were estimated in the left camera frame, therefore, the transformation is between the LiDAR and left camera. The achieved 3D-3D transformation is less accurate than the 3D-2D estimation. Figure 32 shows a comparison between 3D-3D and 3D-2D extrinsic parameters projecting the point cloud on the image. We visually evaluate the transformation and see that 3D-2D based transformation point cloud results in a better projection on the image. We concluded that the best approach

Table 8: Stereo calibration results inside (top) and outside (bottom) the car, with all available distortion models. Calibration is based on 11250 point correspondences.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	**Inside**	-	-	-	-	-	-	-	-
2	T_x**(m)**	-0.96	-0.97	-0.98	-0.98	-0.98	-0.98	-0.98	-0.98
3	T_y**(m)**	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	T_z**(m)**	0.20	0.188	0.11	0.11	0.11	0.12	0.12	0.12
5	R_x**°**	0.68	0.68	8.23	8.23	8.89	4.67	4.66	4.64
6	R_y**°**	22.66	21.67	21.00	21.00	21.33	20.25	20.25	20.27
7	R_z**°**	-1.05	-1.01	0.12	0.12	0.20	0.04	0.04	0.03
8	RMS**(px)**	0.42	0.31	0.21	0.21	0.20	0.19	0.19	0.19
9	**Outside**	-	-	-	-	-	-	-	-
10	T_x**(m)**	-0.96	-0.96	-0.96	-0.96	-0.96	-0.96	-0.96	-0.96
11	T_y**(m)**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	T_z**(m)**	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
13	R_x**°**	0.87	0.88	0.90	0.90	0.96	0.99	1.01	1.04
14	R_y**°**	22.65	22.65	22.66	22.66	22.67	22.68	22.69	22.69
15	R_z**°**	-0.83	-0.83	-0.82	-0.82	-0.80	-0.79	-0.79	-0.78
16	RMS**(px)**	0.15	0.15	0.14	0.14	0.14	0.14	0.14	0.14

Table 9: Camera-LiDAR extrinsic parameters between LiDAR-left camera estimated with both calibration boards for 3D-2D case.

	$\mathbf{R_x}^\circ$	$\mathbf{R_y}^\circ$	$\mathbf{R_z}^\circ$	$\mathbf{t_x}(m)$	$\mathbf{t_y}(m)$	$\mathbf{t_z}(m)$
Chessboard	90.8	-8.8	0.7	0.64	-0.5	-0.65
Custom-board	90.4	-8.5	0.72	0.66	-0.46	-0.66
Merged data points	90.6	-8.5	0.71	0.66	-0.48	-0.66

for camera-LiDAR extrinsic is to use 3D-2D point correspondences because the 3D-3D approach has an extra step (3D board reconstruction) that might introduce noise into the calibration. The extrinsic parameters estimated with 2D-3D point correspondences were used to project the whole LiDAR point cloud in the image

Table 10: Camera-LiDAR extrinsic parameters between LiDAR-right camera estimated with both calibration boards for 3D-2D case.

	$\mathbf{R_x}^\circ$	$\mathbf{R_y}^\circ$	$\mathbf{R_z}^\circ$	$\mathbf{t_x}(m)$	$\mathbf{t_y}(m)$	$\mathbf{t_z}(m)$
Chessboard	92.0	14.1	-0.4	-0.65	-0.48	-0.64
Custom-board	91.7	14.14	-0.37	-0.66	-0.46	-0.63
Merged data points	91.8	14.15	-0.37	-0.66	-0.47	-0.64

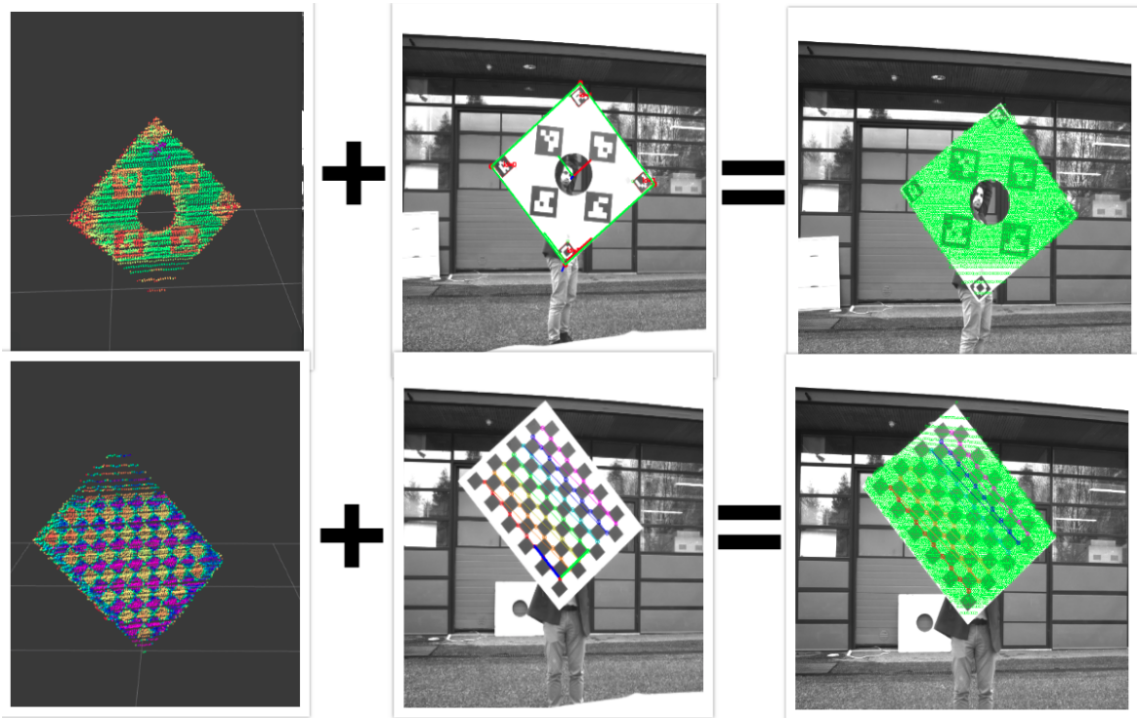

Figure 31: LiDAR points projected on camera space, based on 3D-2D point correspondences.

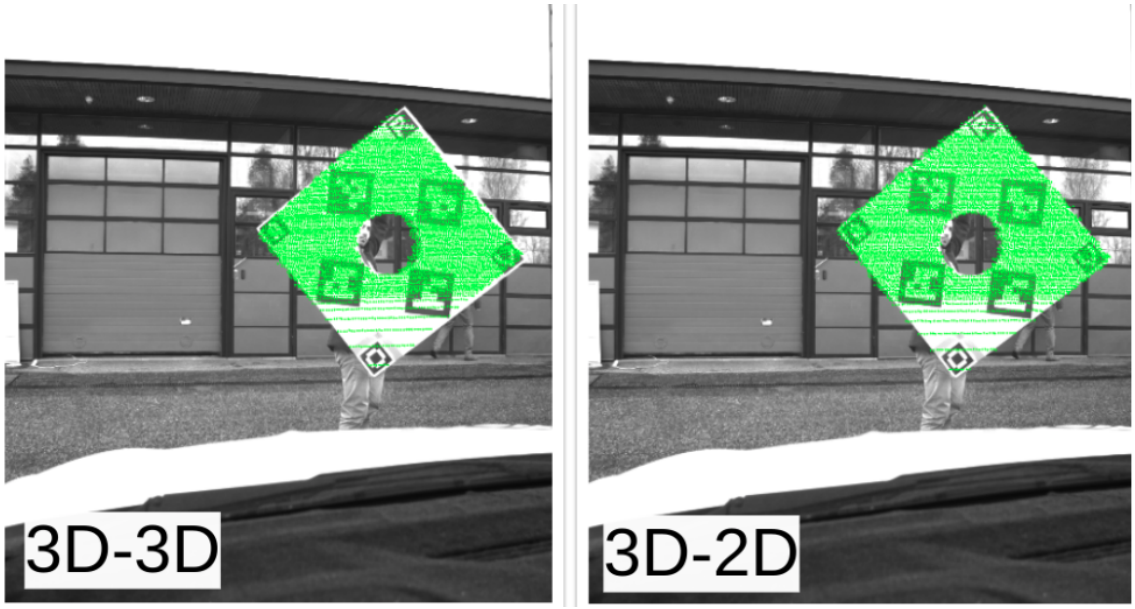

Figure 32: LiDAR to camera projection comparison: left-(3D-3D), right-(3D-2D).

frame. Figure 33 shows the LiDAR and the camera frames on the car, estimated based on 3D-2D point correspondences.

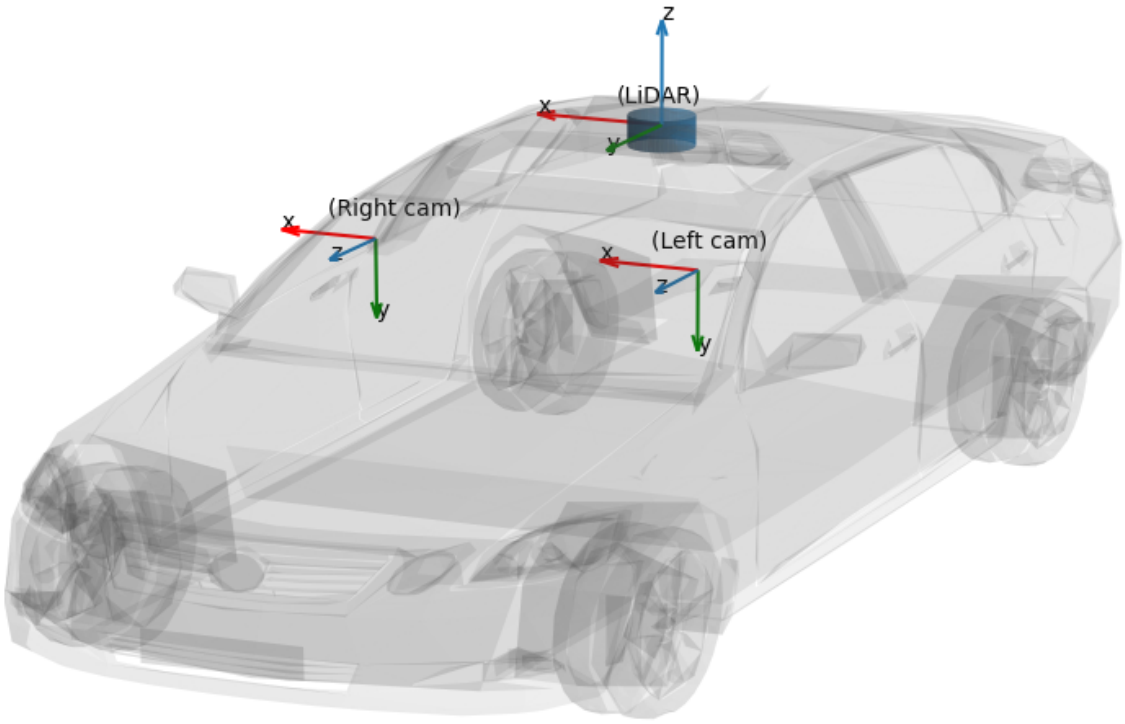

Figure 33: Stereo camera and LiDAR frames extrinsic estimated based on 3D-2D point correspondences. LiDAR is used as the world frame.

Table 11: Camera-LiDAR extrinsic parameters estimated with 3D-3D point correspondences. The transformation is between LiDAR and left camera.

$\mathbf{R_x}^\circ$	$\mathbf{R_y}^\circ$	$\mathbf{R_z}^\circ$	$\mathbf{t_x}(m)$	$\mathbf{t_y}(m)$	$\mathbf{t_z}(m)$
90.73	-5.41	0.35	0.37	-0.47	-0.56

4.3 3D scene reconstruction

In this subsection, we show how the same environment scene is perceived by each sensor. Figure 35 shows the same scene viewed by cameras and LiDAR. Figure 34 shows a comparison between LiDAR-based and camera-based 3D scene reconstruction. Using correlation-based matching, described in Section 2.5.2, we computed the 3D reconstruction of the scene, based on disparity map. The quality of the 3D point cloud is directly bonded to the disparity map. The used block matching technique for disparity computation requires a block size parameter. Smaller block size results

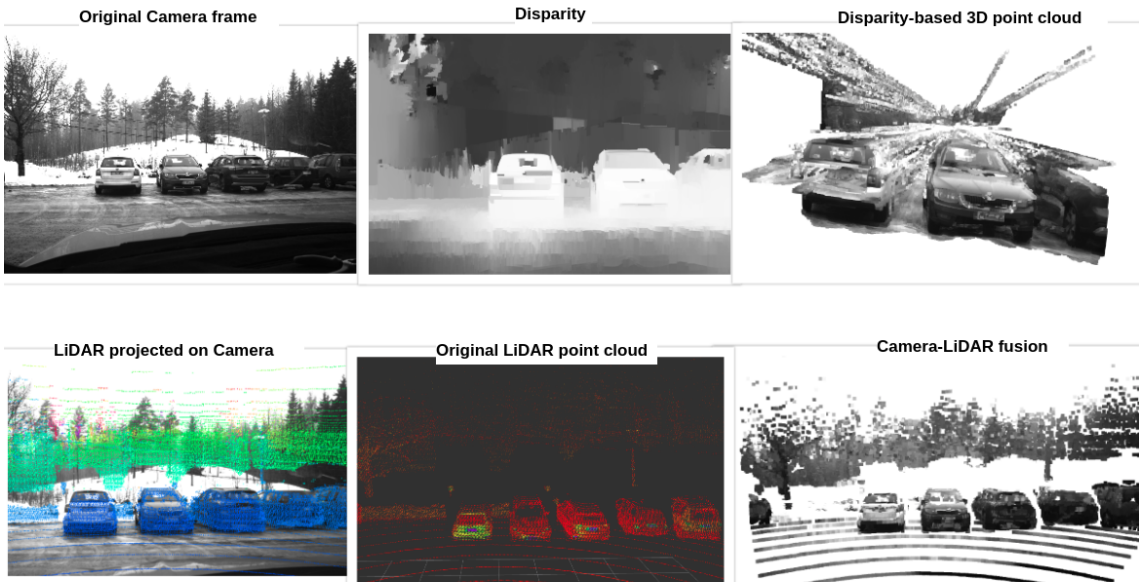

Figure 34: Camera-based and LiDAR-based 3D scene reconstruction. Top - original image, disparity map, camera 3D point cloud based on disparity map. Bottom - LiDAR point cloud projected on camera frame, original point cloud, 3D point scene reconstruction based on camera-LiDAR fusion.

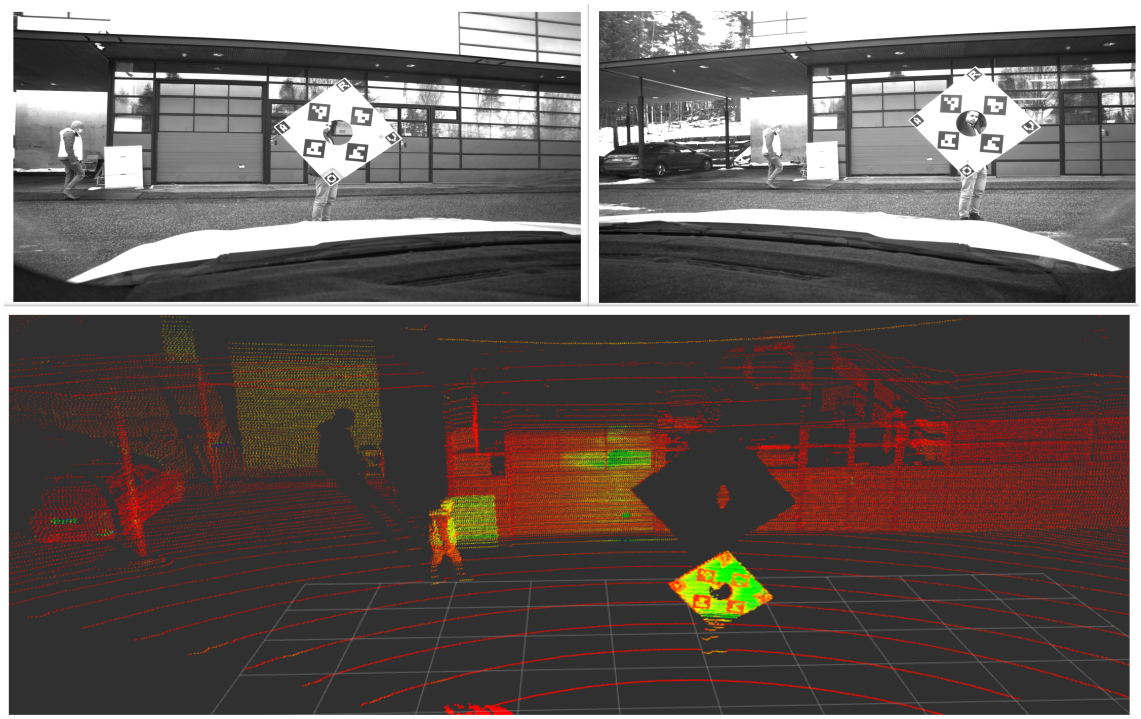

Figure 35: The same scene viewed by cameras (top) and LiDAR (bottom).

in a more detailed scene, however, it introduces noise due to ambiguity (the same block size may have multiple matches) and textureless regions in the scene. A bigger block size may not detect small objects in the scene. In this example, block size of 15 pixel was used. Using camera-LiDAR extrinsic parameters, the LiDAR point cloud was projected on the image frame. Each LiDAR point gets the colour from its pixel projection. The colour of the projected points on the image space is based on point depth. Figure 36 shows the camera-LiDAR fusion.

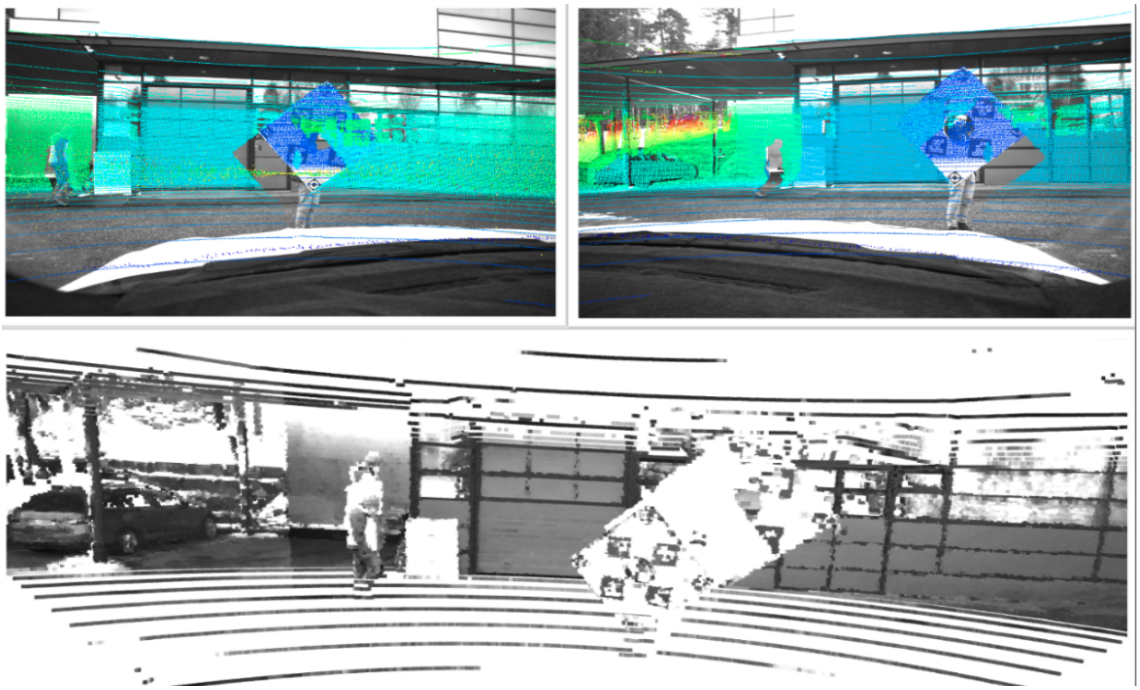

Figure 36: Camera-LiDAR fusion before occlusion removal. Top-point cloud projected on the image frame, the colour of the projected points is based on their depth. Bottom-point cloud fused with image colour.

4.4 Occlusion removal

On the camera frame in Figure 36, we noticed that points that lie on the calibration board do not have the same colour. This is because of sensors occlusion issue described in Section 3.5.3. As we have seen in Figure 33, cameras and LiDAR have different locations, due to this, some LiDAR points cannot be seen by the camera. The occluded points get a wrong colour on the camera-LiDAR projection. We notice this shadow effect on the 3D point cloud, which causes ambiguity. Algorithm 1, presented in Section 3.5.3 was applied to filter occluded points. The number of nearest neighbours and box size parameters must be tuned accordingly, otherwise the Algorithm 1, may suffer from point cloud sparsity. In this thesis we used five nearest neighbours for each point and treat each point as a box of 20×20 pixels. The filtered result is presented in Figure 37.

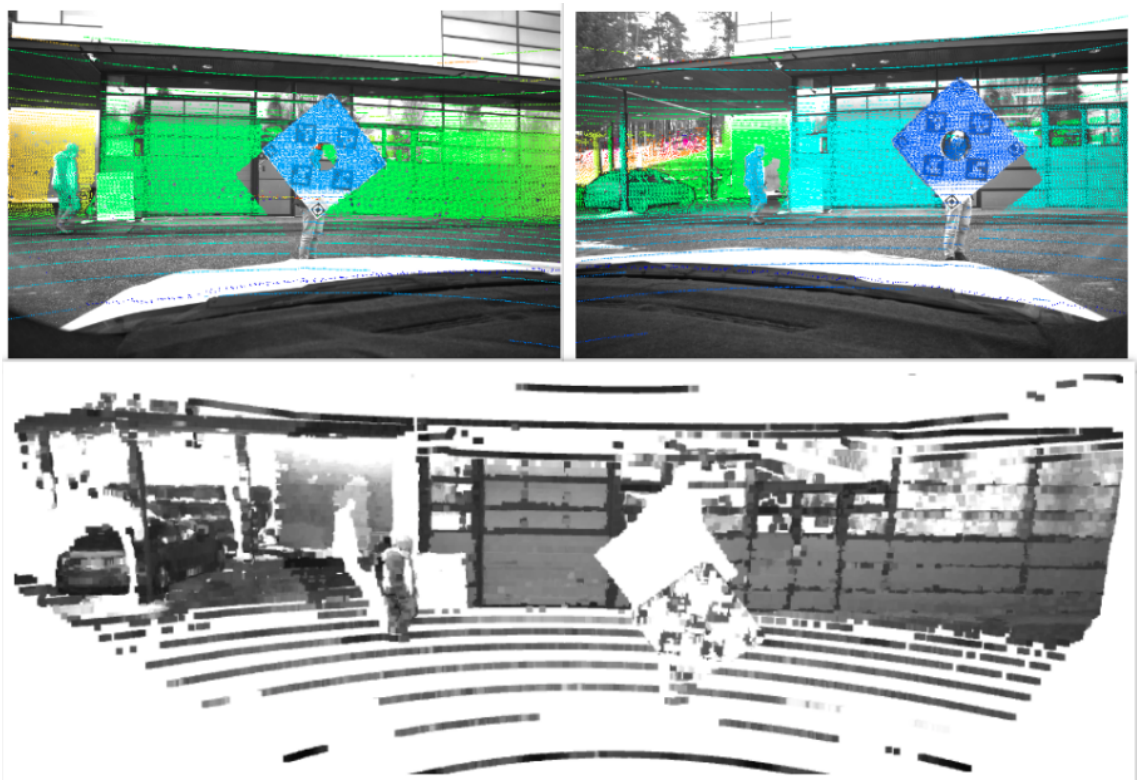

Figure 37: Camera-LiDAR fusion after occlusion removal algorithm. Top - LiDAR points projected on left and right camera. Bottom - LiDAR points fused with camera colour information.

5 Discussion

In this thesis, a camera-LiDAR calibration is performed utilizing mid-level sensor data (i.e., camera and LiDAR feature point correspondences). The windshield effect is tackled by performing mono and stereo camera calibration with cameras installed inside and outside the vehicle. Data sets are collected with several calibration targets, to check how the calibration board influences the results. The extrinsic calibration is based on two types of data point correspondences (2D-3D and 3D-3D). The camera-based and camera-LiDAR-based 3D reconstruction is presented. Usually, ambiguities in sensor fusion is caused by occlusion, therefore, an occlusion handling algorithm is proposed and implemented to filter the ambiguously fused points. In the end, the sensors are calibrated, synchronised, and fused. This thesis presents the following results:

- Calibration parameters estimated based on data sets collected with different calibration targets, are almost the same. We concluded that no calibration board is superior to the other. It is a matter of feature detection technique, however, some calibration targets (e.g., ChArUco) allow the user to easier collect data sets.

- It has been shown that there is a windshield effect on mono camera calibration, which affects the intrinsic parameters, however, it does not affect the transformation between stereo cameras. It increases the focal length, mostly in the y-direction and shifts the principal point down on the xy-axis. The windshield effect also increases the camera calibration re-projection error. We suspect that the extrinsic parameters remain unchanged on inside and outside calibration due to the fact that both cameras are affected by the windshield in a similar manner.

- The use of different distortion models does not bring any significant improvement on camera calibration, however, when cameras are affected by windshield, the complete distortion model can deal with the introduced noise, even if not completely, it can decrease the windshield effect. We suspect that the windshield works as an extra lens for camera calibration that introduces nonlinear noise and changes the light rays' direction. We have seen that the complete distortion model does not completely remove the windshield effect, and we should probably add a light refraction parameter to the distortion polynomial equation. Another idea would be to have ground truth point correspondences with and without a windshield on the vehicle. This would mean that we have to unmount and mount the windshield on the vehicle, keeping the camera and calibration target positions fixed. With ground truth point correspondences, the windshield effect can be handled as an extra parameter in the distortion polynomial equation.

- All sensors have their internal clock, which might drift even if they have the same frequency, therefore, sensor synchronisation is an essential and required step to ensure a precise data transmission. It is preferable to perform hardware-based sensor synchronisation because it is the safest and most stable method,

however, when hardware-based synchronisation is not possible, a temporary solution can be achieved via software-based synchronisation.

- The LiDAR point cloud may have a patchy density, which might cause problems in model fitting. We overcome this issue by using the point cloud margins obtained with convex hull algorithm. Another idea to overcome point cloud sparsity when fitting a model will be to estimate point cloud feature descriptors and use them for fitting.

- It has been shown that the most suitable method for extrinsic calibration is to utilize 3D LiDAR and 2D camera pixel point correspondences. Transformation based on 3D-3D point correspondences tends to be less accurate and also introduces an extra computational step (camera-based 3D point reconstruction) in the calibration pipeline.

- We have shown that camera 3D reconstruction based on correlation feature matching is dense but noisy while LiDAR point cloud is accurate but sparse (the density is dependent on the number of LiDAR beams). The noise in camera-based 3D reconstruction is mostly caused by inaccurate disparity map. To the authors' knowledge, none of the existing camera-based disparity estimation methods is as accurate as LiDAR measurements. However, in applications, where depth is not a crucial feature, camera-based methods might offer acceptable solutions.

- The occlusion problem is crucial in object and obstacle detection for autonomous driving. A pedestrian detected by LiDAR may not be detected by the camera and vice-versa. At this point, occlusion handling can improve sensor fusion setup. In this work, we showed how the occlusion introduces ambiguities in camera-LiDAR fusion, therefore, an occlusion filtering algorithm was proposed to filter the occluded points in camera space. Furthermore, we leverage the stereo camera setup to reconstruct the occluded information between the cameras (i.e., the occluded points on the left camera are reconstructed from the right camera and vice-versa).

- It has been shown that camera disparity-based depth is not accurate enough, while LiDAR does not provide texture information, therefore, camera-LiDAR fusion provides accurate depth and texture information to the system by combining the best of both worlds.

6 Conclusion

Sensor fusion is the ability to gather data and balance the strengths of multiple sensors that result in a more accurate perception model of the environment. 3D range scanners combined with stereo camera vision currently tend to be the most utilized perception system for autonomous vehicles. A camera-LiDAR fusion presumes accurate sensors synchronisation and calibration which benefit autonomous driving safety and robustness. In this thesis, stereo cameras and a LiDAR sensor from a research vehicle platform are fused.

This thesis presents camera-LiDAR calibration techniques and in addition, it shows a comparison between the existing methods and finds the most suitable one. Furthermore, it shows that the windshield affects the camera calibration and demonstrates that its effect can be minimized. Sometimes the quality of data points is more important than the actual estimation algorithms, therefore, we focused on accurate as possible data collection. The acquired results are based on data sets collected with multiple calibration targets and validated with different estimation algorithms. The calibration parameters quality is visually demonstrated by perfectly aligning point clouds.

Two monochrome cameras and Velodyne VLS-128 LiDAR have been used for data collection and experiments. We collected two separate camera data sets to inspect the effect of the windshield on camera calibration. One data set was recorded while the cameras were installed inside the vehicle and the other cameras were installed outside. The data set that was recorded outside the car is not affected by the windshield, therefore, the estimated results from it are used as a basis for comparison. A chessboard and a ChArUco target were used to record camera data. The calibration was performed with all available distortion models and it has been shown that the complete distortion model decreases the windshield effect. Data sets for camera-LiDAR extrinsic calibration were also collected with two calibration targets: a chessboard and a custom board. The original LiDAR point cloud was filtered with the RANSAC plane fitting method. Convex hull algorithm was applied to avoid the data sparsity problem and extract the calibration target shape. A target template model was fitted on the filtered point cloud, to extract the features. Extrinsic calibration was performed using 3D-2D (LiDAR and camera pixels) and 3D-3D (LiDAR and 3D stereo camera) point correspondences and different estimation algorithms. PnP method was used for 3D-2D points and ICP with least squares were used for 3D-3D point correspondences. It has been shown that using 3D-2D point correspondences is the most suitable method to estimate the extrinsic parameters since the 3D-3D method has an extra computational step and the result may be affected by camera-based 3D reconstruction inaccuracies. Besides the calibration, sensor synchronisation was performed using GNSS clock. Synchronisation is an important phase that ensures that sensors provide measurements of the same scene at the same time. The occlusion problem was encountered in camera-LiDAR fusion, which causes ambiguities in data fusion. Therefore, an occlusion removal algorithm was developed and applied to filter out the ambiguities in camera-LiDAR fusion.

Cameras and LiDARs can handle most self-driving perception tasks, such as

localisation, mapping, and obstacle detection, however, none of the sensors alone provides long-term accuracy and safety. The importance of camera-LiDAR fusion is obvious, it improves the robustness and accuracy of the perception system by combining precise depth and texture with colour information. Cameras and LiDARs have complementary strengths and weaknesses, however, their fusion show improvements in performance by combining the best of both worlds. Obstacle detection plays an extremely important role in self-driving since it is directly related to the safety and security of passengers and other traffic participants. Typically, the obstacle detection module suffers from occlusion problem. This thesis offers solutions for stereo camera-LiDAR occlusion handling which filters out the ambiguously fused points and reconstruct the missing information by leveraging stereo setup. The provided stereo camera-LiDAR fusion module can further be used in Visual-LiDAR SLAM problems, and the solution can also be easily extended to any camera and laser systems.

The autonomous research vehicle is embedded with five LiDARs, two monochrome, one thermal and three RGB cameras, therefore, in future work, all cameras and LiDARs should be synchronised and calibrated with the current solution. The development will be continued by fusing camera-LiDAR in an actual driving scenario. Also, in future work, we propose to improve the occlusion removal algorithm, (e.g., estimate the box size for each point separately, and select the number of neighbours automatically), propose a better model to deal with the windshield effect on camera calibration and include an IMU in the calibration pipeline.

References

Ayache, N. 1991. *Artificial vision for mobile robots: stereo vision and multisensory perception.* MIT Press.

Besl, P.J., and N.D. McKay. 1992. "Method for registration of 3-D shapes." In *Sensor Fusion IV: Control Paradigms and Data Structures,* 1611:586–606. International Society for Optics and Photonics.

Bradski, G. 2000. "The OpenCV Library." *Dr. Dobb's Journal of Software Tools.*

Burger, W. 2016. "Zhang's camera calibration algorithm: in-depth tutorial and implementation." *HGB16-05,* 1–6.

Chan, T., and D. Lichti. 2013. "Feature-based self-calibration of Velodyne HDL-32E LiDAR for terrestrial mobile mapping applications." In *The 8th International Symposium on Mobile Mapping Technology, Tainan, Taiwan.*

Chetverikov, D., D. Svirko, D. Stepanov, and P. Krsek. 2002. "The trimmed iterative closest point algorithm." In *Object recognition supported by user interaction for service robots,* 3:545–548. IEEE.

Chiou, G. Nelson. 2017. "Reducing the Variance of Intrinsic Camera Calibration Results in the ROS Camera_Calibration Package." Master's thesis, The University of Texas at San Antonio.

Chojnacki, W., and M. J. Brooks. 2003. "Revisiting Hartley's normalized eight-point algorithm." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (9): 1172–1177.

Claudine, B., G. Rânik, V.C. Raphael, A. Pedro, B.C. Vinicius, A. Forechi, J. Luan, et al. 2021. "Self-driving cars: A survey." *Expert Systems with Applications* 165:113816.

Duane, C. Brown. 1971. "Close-range camera calibration." *Photogramm. Eng* 37 (8): 855–866.

Dubrofsky, E. 2009. "Homography estimation." *Diplomová práce. Vancouver: Univerzita Britské Kolumbie* 5.

Ferrão, J., P. Dias, and J.-R. Neves. 2018. "Detection of Aruco Markers Using the Quadrilateral Sum Conjuncture," edited by Aurélio Campilho, Fakhri Karray, and Bart ter Haar Romeny. Springer International Publishing.

Forsyth, D.A., and J. Ponce. 2012. *Computer Vision: A Modern Approach.* Tsinghua University Press.

Fusiello, A., E. Trucco, and A. Verri. 2000. "A compact algorithm for rectification of stereo pairs." *Machine vision and applications* 12 (1): 16–22.

Gallo, O., R. Manduchi, and A. Rafii. 2011. "CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data." *Pattern Recognition Letters* 32 (3): 403–410.

Gao, X.S., X.R. Hou, J. Tang, and H.F. Cheng. 2003. "Complete solution classification for the perspective-three-point problem." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (8): 930–943.

Georgoulas, C., L. Kotoulas, G. Ch Sirakoulis, I. Andreadis, and A. Gasteratos. 2008. "Real-time disparity map computation module." *Microprocessors and Microsystems* 32 (3): 159–170.

Godard, C., A. Mac, F. Oisin, and J. Gabriel. 2017. "Unsupervised monocular depth estimation with left-right consistency." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 270–279.

Graham, R.L., and F. Yao. 1983. "Finding the convex hull of a simple polygon." *Journal of Algorithms* 4 (4): 324–331.

Guindel, C., J. Beltrán, D. Martín, and F. García. 2017. "Automatic extrinsic calibration for LiDAR-stereo vehicle sensor setups." *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).*

Guo, Y., Y. Gu, and Y. Zhang. 2011. "Invariant Feature Point based ICP with the RANSAC for 3-D Registration." *Information Technology Journal* 10 (2): 276–284.

Gwon, H., S. Lee, M.-W. Seo, K. Yun, W.-Y. Cheong, and S.-J. Kang. 2018. "Charuco Board-Based Omnidirectional Camera Calibration Method." *Electronics* 7.

Hall, M. Andrew. 1999. "Correlation-based feature selection for machine learning."

Hartley, R., and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision.* Cambridge University Press.

Hartley, R. I. 1997. "In defense of the eight-point algorithm." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (6): 580–593.

Hecht, J. 2018. "LiDAR for self-driving cars." *Optics and Photonics News* 29 (1): 26–33.

Hirschmuller, J., and D. Scharstein. 2007. "Evaluation of cost functions for stereo matching." In *IEEE Conference on Computer Vision and Pattern Recognition.*

Horn, B. 1990. "Recovering Baseline and Orientation from Essential Matrix."

IDS GmbH. 2021. *UI-3060CP-M-GL Camera.* AB00604. Rev.2. IDS GmbH.

Lepetit, V., F. Moreno-Noguer, and P. Fua. 2009. "Epnp: An accurate O(n) solution to the PnP problem." *International Journal of Computer Vision* 81 (2).

Lichti, D.S., M. Tsakiri, and A.J. Snow. 2000. "Calibration and testing of a terrestrial laser scanner." *International Archives of Photogrammetry and Remote Sensing* 33 (B5/2; PART 5): 485–492.

Loop, C., and Z. Zhang. 1999. "Computing rectifying homographies for stereo vision." In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1:125–131.

Lourakis, M. 2005. "A brief description of the Levenberg-Marquardt algorithm implemented by Levmar." *Foundation of Research and Technology* 4 (1).

Luettel, T., M. Himmelsbach, and H.-J. Wuensche. 2012. "Autonomous ground vehicles—concepts and a path to the future." *Proceedings of the IEEE* 100.

Luo, R.C., C.-C. Yih, and K.L. Su. 2002. "Multisensor fusion and integration: approaches, applications, and future research directions." *IEEE Sensors Journal* 2 (2): 107–119.

Luong, Q.-T., and O. Faugeras. 1996. "The fundamental matrix: Theory, algorithms, and stability analysis." *International Journal of Computer Vision* 17 (1): 43–75.

Luong, Q.T. 1993. "On determining the fundamental matrix: Analysis of different methods and experimental results." PhD diss., Inria.

Mac, A., C. Godard, F. Oisin, and J. Gabriel. 2019. "Digging into self-supervised monocular depth estimation." In *Proceedings of the IEEE/CVF International Conference on Computer Vision.*

Martins, P. F., H. Costelha, L. C. Bento, and C. Neves. 2020. "Monocular Camera Calibration for Autonomous Driving — a comparative study." *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC),* 306–311.

Marut, A., K. Wojtowicz, and K. Falkowski. 2019. "ArUco markers pose estimation in UAV landing aid system." In *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace),* 261–266.

Miądlicki, K., and M. Saków. 2019. "LiDAR Based System for Tracking Loader Crane Operator," 406–421.

Molebny, V., G. Kamerman, and O. Steinvall. 2010. "Laser radar: from early history to new trends." In *Electro-Optical Remote Sensing, Photonic Technologies, and Applications IV.* International Society for Optics and Photonics.

Muhammad, N., and S. Lacroix. 2010. "Calibration of a rotating multi-beam LiDAR." *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 5648–5653.

Paden, B., M. Čáp, S.Z. Yong, D. Yershov, and E. Frazzoli. 2016. "A survey of motion planning and control techniques for self-driving urban vehicles." *IEEE Transactions on Intelligent Vehicles* 1 (1): 33–55.

Putkiranta, P. 2020. “Geometric calibration of rotating multi-beam LiDAR systems” [in English]. Master’s thesis, Aalto University. School of Engineering.

Quigley, Morgan, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, et al. 2009. “ROS: an open-source Robot Operating System.” In *ICRA Workshop on Open Source Software.*

Scaramuzza, D., and F. Fraundorfer. 2011. “Tutorial: visual odometry.” *IEEE Robotics and Automation Magazine* 18 (4): 80–92.

Schneider, S., M. Himmelsbach, T. Luettel, and H. Wuensche. 2010. “Fusing vision and LiDAR - Synchronization, correction and occlusion reasoning.” In *IEEE Intelligent Vehicles Symposium,* 388–393.

Shapiro, R. 1978. “Direct linear transformation method for three-dimensional cinematography.” *Research Quarterly. American Alliance for Health, Physical Education and Recreation* 49 (2): 197–205.

Szabo, Fred E. 2015. “Triple dot product.” In *The Linear Algebra Survival Guide,* edited by F.E. Szabo, 393–410. Boston: Academic Press.

Szeliski, R. 2010. *Computer Vision: Algorithms and Applications.* Springer Science & Business Media.

Thrun, S., W. Burgard, and D. Fox. 2005. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press.

Velodyne. 2018. *VLS-128.* Rev. A 63-9483. Velodyne.

Wang, J, F. Shi, J. Zhang, and Y. Liu. 2008. “A new calibration model of camera lens distortion.” *Pattern Recognition* 41 (2): 607–615.

Wang, Y., W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger. 2019. “Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving.” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 8445–8453.

Yang, J., H. Li, and Y. Jia. 2014. “Optimal essential matrix estimation via inlier-set maximization.” In *European Conference on Computer Vision,* 111–126.

Yaniv, Z. 2010. “Random sample consensus (RANSAC) algorithm, a generic implementation.” *Imaging.*

Yousif, K., A. Bab-Hadiashar, and R. Hoseinnezhad. 2015. “An overview to visual odometry and visual SLAM: Applications to mobile robotics.” *Intelligent Industrial Systems* 1 (4): 289–311.

Zhang, J. 2000. “A flexible new technique for camera calibration.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (11): 1330–1334.

Zhang, J., and S. Singh. 2015. "Visual-LiDAR odometry and mapping: Low-drift, robust, and fast." *IEEE International Conference on Robotics and Automation (ICRA),* 2174–2181.

Zou, L., and Y. Li. 2010. "A method of stereo vision matching based on OpenCV." In *International Conference on Audio, Language and Image Processing,* 185–190. IEEE.

7 Appendix

In this section, we present the full results for camera calibration, intrinsic matrix and distortion patameters for all available distortion models.

Table 12: Full intrinsic **Inside left** camera calibration based on 11596 points correspondences. Values are in pixels.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	**fx**	1372.01	1371.56	1373.73	1373.73	1373.10	1366.83	1369.43	1366.59
2	**fy**	1377.99	1377.56	1379.77	1379.77	1379.32	1370.16	1375.54	1369.78
3	**px**	966.343	966.10	1059.35	1059.35	1057.06	970.89	1033.16	965.53
4	**py**	589.17	589.04	524.34	524.34	522.86	599.64	632.67	602.0
5	**sk**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	**k1**	-0.144	-0.27	-0.13	-0.13	1.394	-0.137	-0.168	-0.205
7	**k2**	0.10	0.66	0.09	0.09	0.344	0.08	0.461	5.17
8	**p1**	-0.00	-0.00	-0.02	-0.02	-0.021	-0.038	-0.006	-0.038
9	**p2**	0.00	0.001	0.024	0.024	0.023	-0.02	-0.0	-0.022
10	**k3**	-0.008	0.110	-0.02	-0.02	0.003	0.006	0.008	0.008
11	**k4**	—	-2.65	0.0	0.0	1.52	0.0	-1.52	-1.92
12	**k5**	—	6.12	0.0	0.0	3.59	0.0	4.21	4.80
13	**k6**	—	12.58	0.0	0.0	3.84	0.0	9.84	9.468
14	**s1**	—	0.0	-0.03	-0.03	-0.03	0.02	0.0	0.02
15	**s2**	—	0.0	0.006	0.006	0.006	0.0008	0.0	0.0
16	**s3**	—	0.0	0.02	0.02	0.02	0.03	0.0	0.03
17	**s4**	—	0.0	-0.0	-0.0	-0.0	0.0	0.0	0.0
18	**tx**	—	0.0	—	—	—	0.07	0.03	0.079
19	**ty**	—	0.0	—	—	—	-0.045	-0.04	-0.04
20	**Error**	0.43	0.42	0.36	0.36	0.36	0.33	0.40	0.33

Table 13: Full **Outside left** camera calibration based on 11813 points correspondences. Values are in pixels.

	params	ST	RAT	THP	TIL	RAT+THP	THP+TIL	RAT+TIL	CMP
1	**fx**	1367.32	1367.0	1367.45	1367.45	1367.12	1367.35	1367.05	1367.04
2	**fy**	1367.47	1367.159	1367.58	1367.58	1367.25	1367.54	1367.20	1367.22
3	**px**	966.22	966.23	975.53	975.53	976.74	965.57	967.59	965.6
4	**py**	604.29	604.318	610.13	610.13	611.73	607.35	607.28	607.39
5	**sk**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	**k1**	-0.14	-0.39	-0.142	-0.142	-0.174	-0.142	-0.33	-0.398
7	**k2**	0.094	0.21	0.096	0.096	0.186	0.09	0.181	0.20
8	**p1**	-0.0	-0.0	0.0	0.0	0.0	-0.0	-0.0	-0.0
9	**p2**	-6.59e-05	-6.57e-05	0.0	0.0	0.	-0.	-0.	-0.
10	**k3**	-0.	-0.0025	-0.01	-0.01	-0.0271	-0.00	-0.002	-0.025
11	**k4**	—	-0.26	0.0	0.0	-0.03	0.0	-0.20	-0.26
12	**k5**	—	0.149	0.0	0.0	0.1	0.0	0.12	0.13
13	**k6**	—	-2.7	0.0	0.0	-2.92	0.0	-2.8	-2.75
14	**s1**	—	0.0	-0.	-0.	-0.	0	0.0	0.0
15	**s2**	—	0.0	0.0	0.0	0.0	0.03	0.0	0.0
16	**s3**	—	0.0	-0.	-0.	-0.0	-0.0	0.0	-0.0
17	**s4**	—	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	**tx**	—	0.0	—	—	—	0.0	0.0	0.0
19	**ty**	—	0.0	—	—	—	-0.0	-0.0	-0.0
20	**Error**	0.20	0.19	0.20	0.20	0.19	0.20	0.19	0.19

Table 14: Full **Inside right** camera calibration based on 11668 points correspondences. Values are in pixels.

	params	ST	RAT	THP	TIL	RAT+THP	THP+TIL	RAT+TIL	CMP
1	**fx**	1367.58	1367.76	1374.1	1374.1	1373.9	1368.81	1372.14	1369.16
2	**fy**	1373.75	1373.93	1382.30	1382.30	1382.22	1375.82	1387.12	1376.04
3	**px**	955.74	955.68	863.28	863.28	863.68	990.16	711.57	988.88
4	**py**	601.97	601.942	502.07	502.07	501.57	691.87	653.47	691.36
5	**sk**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	**k1**	-0.149	-3.54	-0.13	-0.13	0.52	-0.14	-0.19	-0.58
7	**k2**	0.11	5.00	0.0	0.0	0.87	0.12	6.455	2.5
8	**p1**	-0.0	-0.0	-0.02	-0.02	-0.02	0.048	-0.005	0.04
9	**p2**	0.0	0.0	-0.02	-0.02	-0.0	-0.01	0.018	-0.01
10	**k3**	-0.03	10.63	-0.027	-0.027	0.29	-0.051	2.003	7.82
11	**k4**	—	-3.3901	0.0	0.0	0.6	0.0	-0.0	-0.43
12	**k5**	—	4.30	0.0	0.0	0.8	0.0	6.4	2.24
13	**k6**	—	12.11	0.0	0.0	0.3	0.0	2.8	8.69
14	**s1**	—	0.0	0.03	0.03	0.03	0.01	0.0	0.01
15	**s2**	—	0.0	-0.	-0.	-0.0	0.	0.0	0.0
16	**s3**	—	0.0	0.0	0.0	0.0	-0.055	0.0	-0.05
17	**s4**	—	0.0	-0.005	-0.005	-0.005	0.004	0.0	0.004
18	**tx**	—	0.0	—	—	—	-0.06	0.032	-0.06
19	**ty**	—	0.0	—	—	—	-0.06	0.169	-0.060
20	**Error**	0.52	0.51	0.34	0.34	0.34	0.31	0.37	0.31

Table 15: Full **Outside right** camera calibration based on 11927 points correspondences. Values are in pixels.

	params	**ST**	**RAT**	**THP**	**TIL**	**RAT+THP**	**THP+TIL**	**RAT+TIL**	**CMP**
1	**fx**	1367.51	1367.27	1367.58	1367.58	1367.35	1367.5	1367.34	1367.33
2	**fy**	1367.49	1367.24	1367.5	1367.5	1367.32	1367.56	1367.3	1367.37
3	**px**	953.09	953.061	954.315	954.315	954.309	960.84	947.92	960.80
4	**py**	610.596	610.587	616.17	616.17	616.14	615.79	613.34	615.77
5	**sk**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	**k1**	-0.14	0.017	-0.145	-0.145	0.03	-0.144	-0.02	0.03
7	**k2**	0.104	0.6	0.104	0.104	0.54	0.10	0.39	0.5
8	**p1**	-0.	-0.	0.	0.	0.	-0.	-0.	-0.
9	**p2**	0.	0.0	0.	0.	0.	0.	0.	0.
10	**k3**	-0.014	-1.05	-0.014	-0.014	-1.28	-0.011	-1.60	-0.97
11	**k4**	—	0.157	0.0	0.0	0.172	0.0	0.11	0.172
12	**k5**	—	0.636	0.0	0.0	0.5	0.0	0.35	0.505
13	**k6**	—	-1.11	0.0	0.0	-1.3	0.0	-1.72	-1.03
14	**s1**	—	0.0	-0.0	-0.0	-0.0	-0.0	0.0	-0.
15	**s2**	—	0.0	0.	0.	0.	0.	0.0	0.
16	**s3**	—	0.0	-0.	-0.	-0	-0.	0.0	-0.
17	**s4**	—	0.0	0.	0.	0.	0.	0.0	0.
18	**tx**	—	0.0	—	—	—	0.	0.	0.
19	**ty**	—	0.0	—	—	—	0.0	0.	0.
20	**Error**	0.172	0.171	0.171	0.171	0.171	0.171	0.17	0.17